2-1-1996

The Interface and characterization of an LFESR spectrometer and preliminary study to detect ultrasound induced radicals by LFESR spectroscopy

William Ryan

The interface and characterization of an LFESR spectrometer and a preliminary study to detect ultrasound induced radicals by LFESR spectroscopy.

William J. Ryan

February 1, 1996

A Thesis
submitted in partial fulfillment of the requirements for the degree of
Master of Science in Chemistry.

Approved: Joseph Homok
Thesis Advisor

G. A. Takacs
Department Head

Department of Chemistry
Rochester Institute of Technology
Rochester, NY 14623-5603

Copyright Release Form

The interface and characterization of an LFESR spectrometer and a preliminary study to detect ultrasound induced radicals by LFESR spectroscopy.

I William John Ryan hereby grant permission to the Wallace Memorial Library of RIT to reproduce my thesis in whole or in part. Any use will not be for commercial use or profit.

Signature _____

Date 2-1-96

Abstract

A modular low frequency electron spin resonance (LFESR) spectrometer has been interfaced, characterized, and applied in a test to detect ultrasound-induced free radicals with low frequency ESR spectroscopy. The modular system, which operates at ~250 MHz, was interfaced with a 486DX2 66 MHz PC, using a multifunction PC compatible interface board to facilitate integrated software control of the modular components. The operating software was programmed in Microsoft Visual C++ to execute in the Windows operating environment. Computer control of the magnet was calibrated, using the 2,5-di-tert-butyl parabenzosemiquinone anion radical as a g-factor and hyperfine splitting standard. Preparation of the semiquinone standard from 2,5-di-tert-butylhydroquinone is described. The spectrometer was characterized, using an aqueous solution of potassium peroxylamine disulfonate [$K_2NO(SO_3)_2$], (Fremy's salt) in 10% sodium hydroxide, as a spin standard. The optimized spectrometer was found to have a line width dependent detection limit of 4.52×10^{21} spins T^{-1} A study was carried out to test the ability of the LFESR spectrometer to detect free radicals spin trap adducts generated from the 20 kHz, 120 watt / cm^2 sonication of aqueous solutions of the nitrone spin trap, 5,5-dimethyl-1-pyroline N-oxide, (DMPO). No radicals were detected in the tests. Fricke dosimetric solutions were used to confirm the generation of radicals. Attempts to measure the optical absorbance of DMPO-OH were unsuccessful in confirming the formation of DMPO-OH radical adducts. Future variations of this study, utilizing integrated ultrasound-sample probe assemblies are suggested to achieve detectable steady state concentrations of DMPO-H and DMPO-OH spin trap adducts, since rapid decay of the adducts to undetectable concentrations by bimolecular and unimolecular processes seem to be the principle problem to deal with at the conclusion of the present study.

List of Figures, Tables, etc.

Part I
Development of an LFESR computer interface subsystem

Figures:

Equations:

Part II
Performance characterization of the LFESR spectrometer

Figures:

Table

Part III
LFESR detection of free radicals generated by ultrasound - a preliminary study

Figures:

Tables:

Reactions:

Equations:

List of Abbreviations

A	Hyperfine splitting constant, in units of Hz.
A/D	Analog-to-digital.
a_H^{β}	Hyperfine splitting constant due to hydrogen nuclei in the beta position to the radical center, in units of T.
a_N	Hyperfine splitting constant due to a nitrogen nucleus, in units of T.
A_{pp}	Peak-to-peak derivative amplitude.
β	Bohr magneton, 9.27402×10^{-24} J/T.
β_n	Nuclear magneton, 5.05082×10^{-27} J/T.
B	Applied magnetic field.
B_1	Magnetic component of the incident electromagnetic radiation, that is perpendicular to the direction of the applied magnetic field, B.
BCD	Binary-coded-decimal.
$B_{lattice}$	Magnetic field that arises from tumbling solvent molecules in the lattice.
B_{loc}	Local magnetic field, arising from unpaired nuclei.
B_{pp}	Peak-to-peak derivative amplitude.
B_{res}	Applied magnetic field at which ESR occurs.
ΔE	The change in energy that is associated with a transition between possible Zeeman energy levels.
D/A	Digital-to-analog.
dB	Decibel. Equal to $10 \text{Log}_{10} (P/P_o)$.
DMPO	5,5-dimethyl-1-pyroline N-oxide. A nitroxide spin trap.
ESR	Electron spin resonance.
Fremy's Salt	Potassium peroxylamine disulfonate, $[K_2NO(SO_3)_2]$.
g	g factor.
g_e	g factor of a free electron, 2.0023.
g_n	nuclear g factor.
h	Planck's constant, 6.62608×10^{-34} Js.
·H	The hydrogen radical.
I	Nuclear spin quantum number.
m_I	Z-component of the nuclear spin quantum number.
m_s	Z-component of the electronic spin quantum number.
n	Number of samples averaged.
·OH	The hydroxyl radical.
P	RF power.
P_o	Reference RF power.
RF	Radio frequency.
S	Electronic spin quantum number.
SNR	Signal-to-noise ratio.
STS	Single-turn-solenoid.
ν	Frequency of incident electromagnetic radiation.

To Jennifer.

Table of Contents

Part I

Development of an LFESR Computer Interface Subsystem

1.1 Introduction

There has been a recent increase in interest in electron spin resonance (ESR) spectroscopy at low frequencies for the study of free radicals in biological systems (Halpern *et al.*, 1989; Lurie *et al.*,1988). Conventional ESR spectrometers typically operate at a frequency of 9.5 GHz (Wertz and Bolton, 1972). A potential disadvantage of the 9.5 GHz operating frequency with respect to biological studies involving aqueous samples, is that 9.5 GHz radiation is strongly absorbed by water

(Montgomery, 1947). This restricts aqueous sample volumes to fractions of a milliliter, seriously compromising the potential application of conventional ESR spectroscopy to *in vivo* studies.

In response to the increased interest in low-frequency electron spin resonance (LFESR) spectroscopy for the study of radicals in biological systems, as well as the lack of commercial availability of LFESR spectrometers, a modular LFESR spectrometer has recently been developed (Hornak *et al.*, 1991). The spectrometer is capable of operating at frequencies between 100 and 400 MHz, and and has a detectable aqueous sample volume of 15 mL. At 200 MHz, the system is reported to have a line-width-dependent detection limit of 8×10^{21} spins T^{-1} (Hornak *et al.*, 1991). This spectrometer was interfaced, using an IBM XT computer in conjunction with digital and serial interface components. The system software was developed to run in a DOS™ operating environment. A natural extension of the development of this system is to develop a Windows™ based operating system. This would allow the end-user of the system to perform a wide variety of data processing operations, utilizing the expansive host of Windows-based applications currently available to PC users.

The goal set forth here is to develop a new computer interface subsystem for the existing modular LFESR spectrometer, that will allow the system to operate within the Windows operating environment. This goal is met by implementing a state-of-the-

to be part of a new 486 DX2 66 MHz PC compatible which replaces the current IBM XT computer. The interface is to be controlled by a Windows-based software package developed utilizing Microsoft Visual C++.

1.2 Background and Theory

The LFESR spectrometer that is being interfaced was originally developed from several modular components. Most of these components are commercially available, and were purchased as off-the-shelf items. These components are listed in Appendix A. Some of the components were custom fabricated as required by the specific design of the spectrometer. These components include the RF-bridge, the magnet, and the sample probe assembly. The objective here is to design and implement a computer controlled interface between the various modular components of the LFESR spectrometer. Section 1.2.1 is intended to provide a basic description of ESR theory, while section 1.2.2 gives a detailed description of the modular LFESR spectrometer being interfaced in this project. For the interested reader, a general description of ESR instrumentation is given in Appendix E.

1.2.1 *Electron Spin Resonance (ESR) Spectroscopy*

Electron spin resonance (ESR) spectroscopy is an instrumental method of studying atoms or molecules that have unpaired electrons. Species that have unpaired electrons are generally called radicals, and have magnetic dipoles associated with the unpaired electrons. ESR spectroscopy is based on the resonant coupling of the magnetic component of incident radiation with the magnetic dipoles of unpaired electrons in a sample.

Magnetic dipoles associated with electrons arise from net orbital angular momentum, net spin angular momentum, or a combination of both. As a consequence of the quantization of angular momentum, each magnetic dipole in a sample will align in a position that is either parallel or antiparallel with the direction of an applied static magnetic field. As defined, the energy of the magnetic dipole will be lower for the parallel orientation, while higher for the antiparallel orientation. Thus, for a given applied magnetic field strength, B, there will be two possible energy states for the magnetic dipole of an electron to be in.

The energy of each of the two possible states of the magnetic dipole of an electron, has a value of $m_s g\beta B$. m_s is the z-component of the electronic spin quantum number and has the values of $\pm s$, where s is the electronic spin quantum number and has a value of 1/2. g is a constant called the g-factor, which is 2.0023 for a free electron. β is the Bohr magneton, and has a value of 9.27402×10^{-24} J/T. The energy

of the upper energy state of the magnetic dipole is defined here as $(+1/2)g\beta B$, while the energy of the lower energy state of the magnetic dipole is defined as $(-1/2)g\beta B$. These energy states are called the Zeeman levels. A transition between Zeeman levels at a given magnetic field strength, B, involves a change in energy, ΔE, equal to $g\beta B$, as defined by Equation 1-1.

$$\Delta E = (+1/2)g\beta B - (-1/2)g\beta B = g\beta B \qquad \textbf{Equation 1-1}$$

As indicated by Equation 1-1, there is a discrete, quantized separation between the Zeeman levels. This separation, ΔE, is directly proportional to the strength of the applied magnetic field, B. Furthermore, the Zeeman levels are degenerate in the absence of an applied magnetic field.

Electron spin resonance occurs when ΔE of the Zeeman energy levels exactly matches the energy of incident photons of monochromatic radiation, causing resonant excitations of the dipole between energy levels. Photons of incident radiation have energies equal to $h\nu$. h is Planck's constant, and has a value of 6.62608×10^{-34} Js. ν is the frequency of the radiation, and has units of s^{-1} The fundamental condition of electron spin resonance is expressed in Equation 1-2,

$$\Delta E = h\nu = g\beta B_{res} \qquad \textbf{Equation 1-2}$$

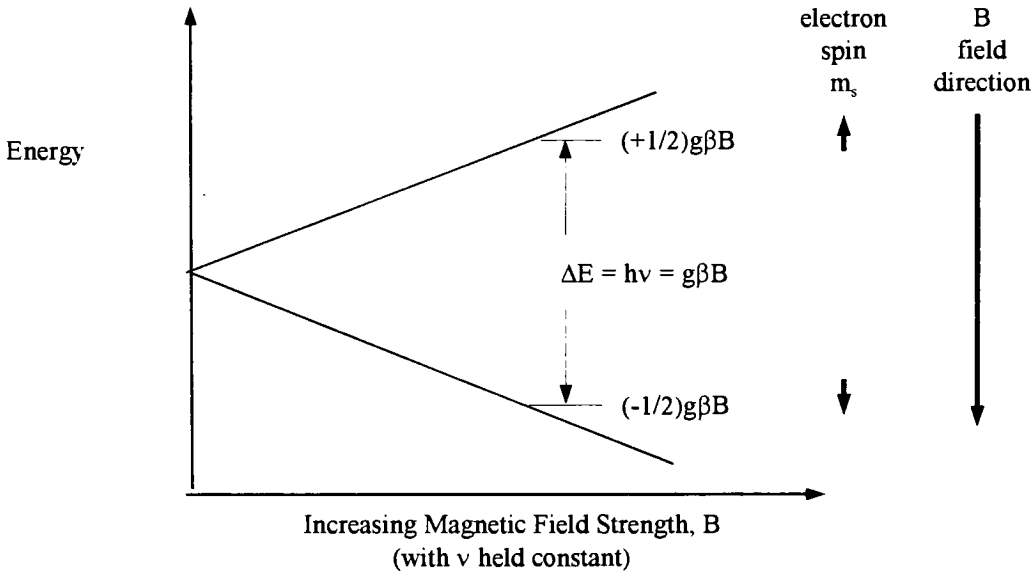
where, B_{res} is the applied magnetic field strength at which resonance is met.

ESR spectroscopy involves sweeping the applied magnetic field, with the frequency of the incident radiation held constant. Resonant absorption of the source radiation is detected as the applied field passes through the resonant condition or conditions for the sample. For example, a free electron will experience resonant absorption of 250 MHz source radiation as the magnetic field is swept through a strength of 8.92 mT, as illustrated in Figure 1-1.

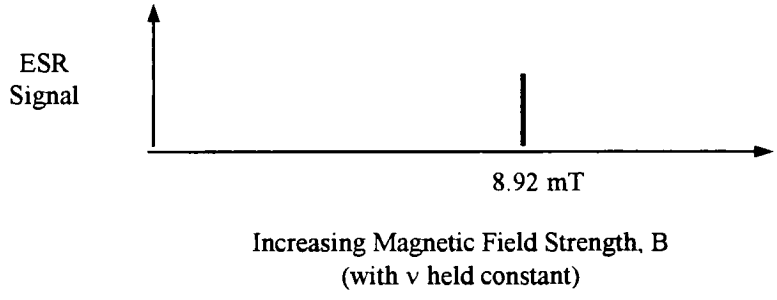
Just as the the magnetic dipoles of unpaired electrons in a sample will align with or against the applied magnetic field, the unpaired magnetic dipoles of unpaired nuclei will also align in a parallel or antiparallel orientation. The local magnetic fields that arise from unpaired nuclei, B_{loc}, superimpose with the applied magnetic field, B. The local magnetic fields, B_{loc}, have strengths of am_I. a is called the hyperfine splitting constant, and is has units of magnetic field strength. m_I is the z-component of the nuclear spin quantum number, and has values of (-I, -I + 1, ..., I), where I is the nuclear spin quantum number of the nucleus being considered. The possible values of B_{res}, are given by Equation 1-3.

$$B_{res} = B + am_I \qquad \textbf{Equation 1-3}$$

This results in the splitting of Zeeman energy levels, and is referred to as hyperfine splitting. For example, consider the hydrogen atom, which consists of a single unpaired electron (s = 1/2), and a single unpaired proton (I = 1/2). The electron has



(a)



(b)

Figure 1-1 (a) An energy diagram for an unpaired electron. (b) Line spectra of an unpaired electron experiencing ESR absorption of 250 MHz radiation at a field strength of 8.8 mT, as calculated by Equation 1-2.

two possible spin states ($m_s = \pm 1/2$), and the proton has two possible spin states ($m_I = \pm 1/2$). The upper and lower Zeeman energy levels of the single unpaired electron are each split as a result of the two possible spin state orientations of the interacting proton. The split Zeeman energy levels of the hydrogen atom, and the stick representation of the resulting ESR spectrum, are illustrated in Figure 1-2.

Although hyperfine splitting results in 4 separate spin energy levels for the hydrogen atom, ESR transitions are only allowed where $\Delta m_s = \pm 1$, and $\Delta m_I = 0$. Therefore, there are only two ESR absorption peaks for the hydrogen atom, as indicated in Figure 1-2.

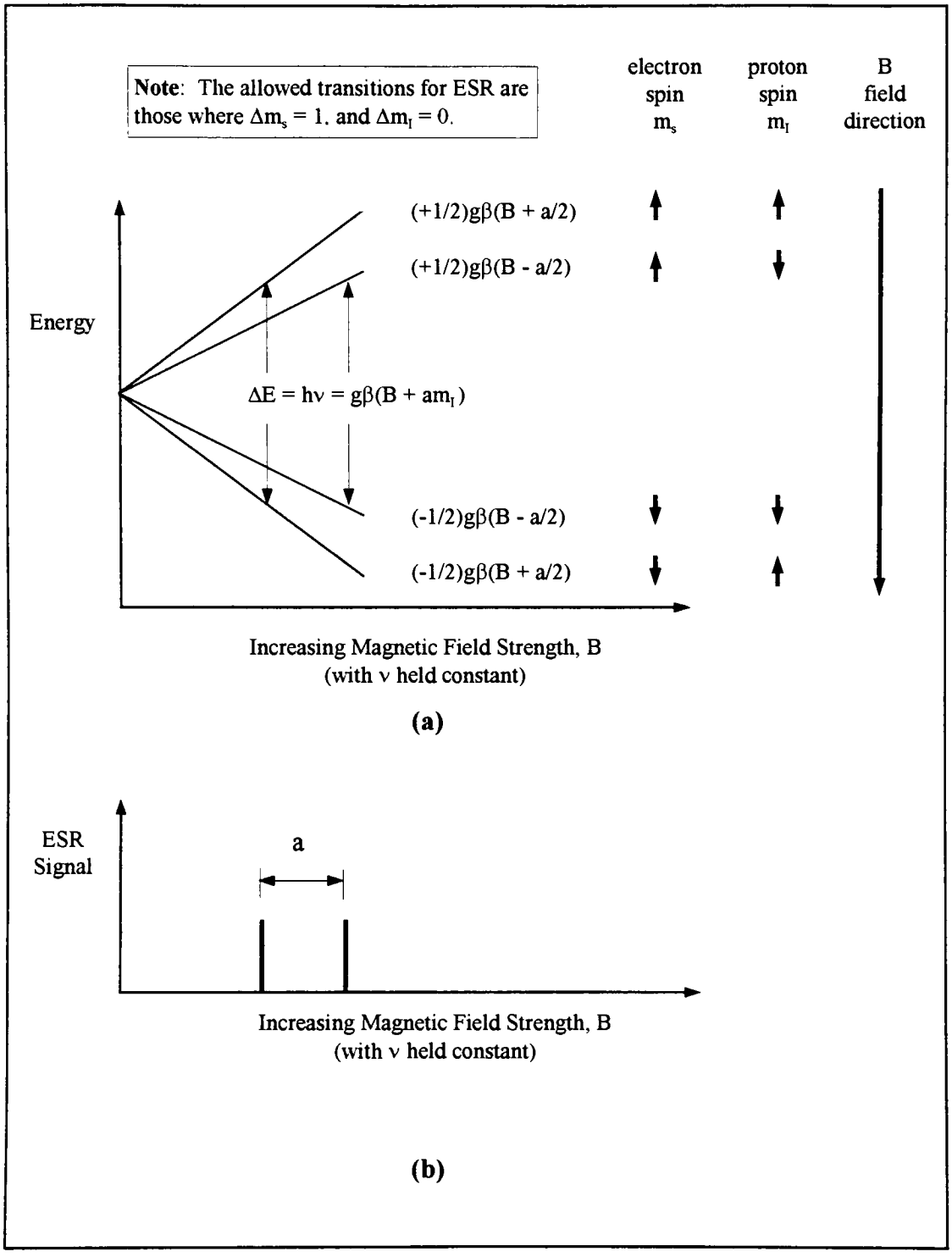


Figure 1-2 (a) An energy diagram for atomic hydrogen. (b) Stick spectra of atomic hydrogen, showing a doublet peak. The doublet is composed of two lines of equal intensity, corresponding to the two equally probable spin state energy transitions.

If there are multiple magnetic nuclei in a radical, each nuclei contributes to the hyperfine splitting. If the nuclei are equivalent (i.e. if they are in the same kind of atom, and they are located in the same relative position on the radical with respect to the unpaired electron), n + 1 hyperfine lines will result. The equivalent hyperfine splittings result in degenerate spin energy states, giving rise to a binomial intensity distribution in the observed peaks of the ESR spectrum. The relative intensities of the peaks are given by Pascal's triangle, as shown in Figure 1-3.

An example of hyperfine splitting by n equivalent nuclei is given in Figure 1-4, for the methyl radical ($\cdot CH_3$), which contains a single unpaired electron (s = 1/2) and 3 equivalent protons (I = 1/2). The hyperfine splitting constant, a, for the methyl radical has a value of ~2.5 mT.

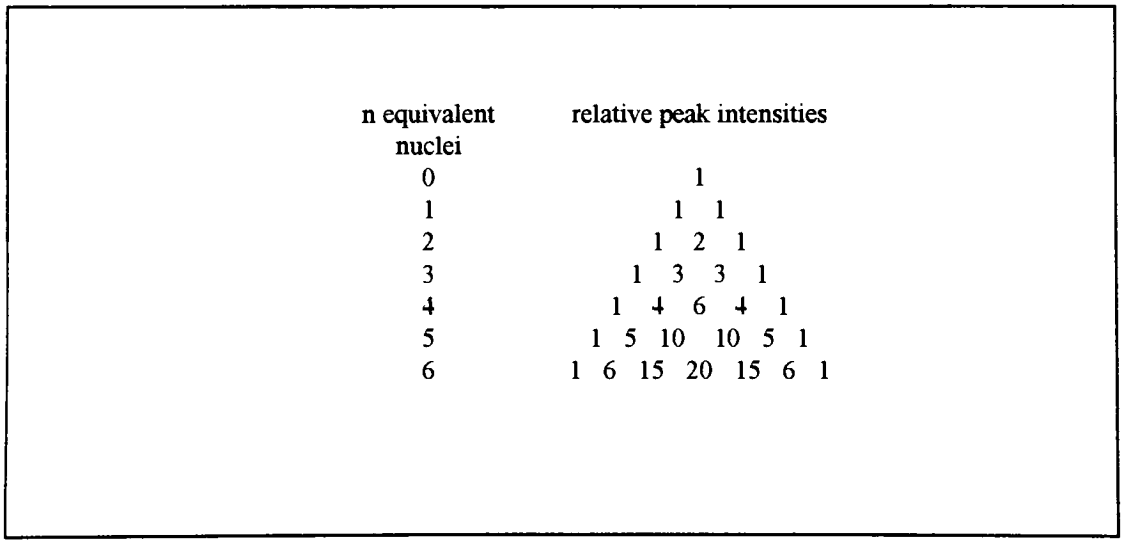


Figure 1-3 Relative hyperfine peak intensities resulting from n equivalent nuclei. Based on Pascal's triangle.

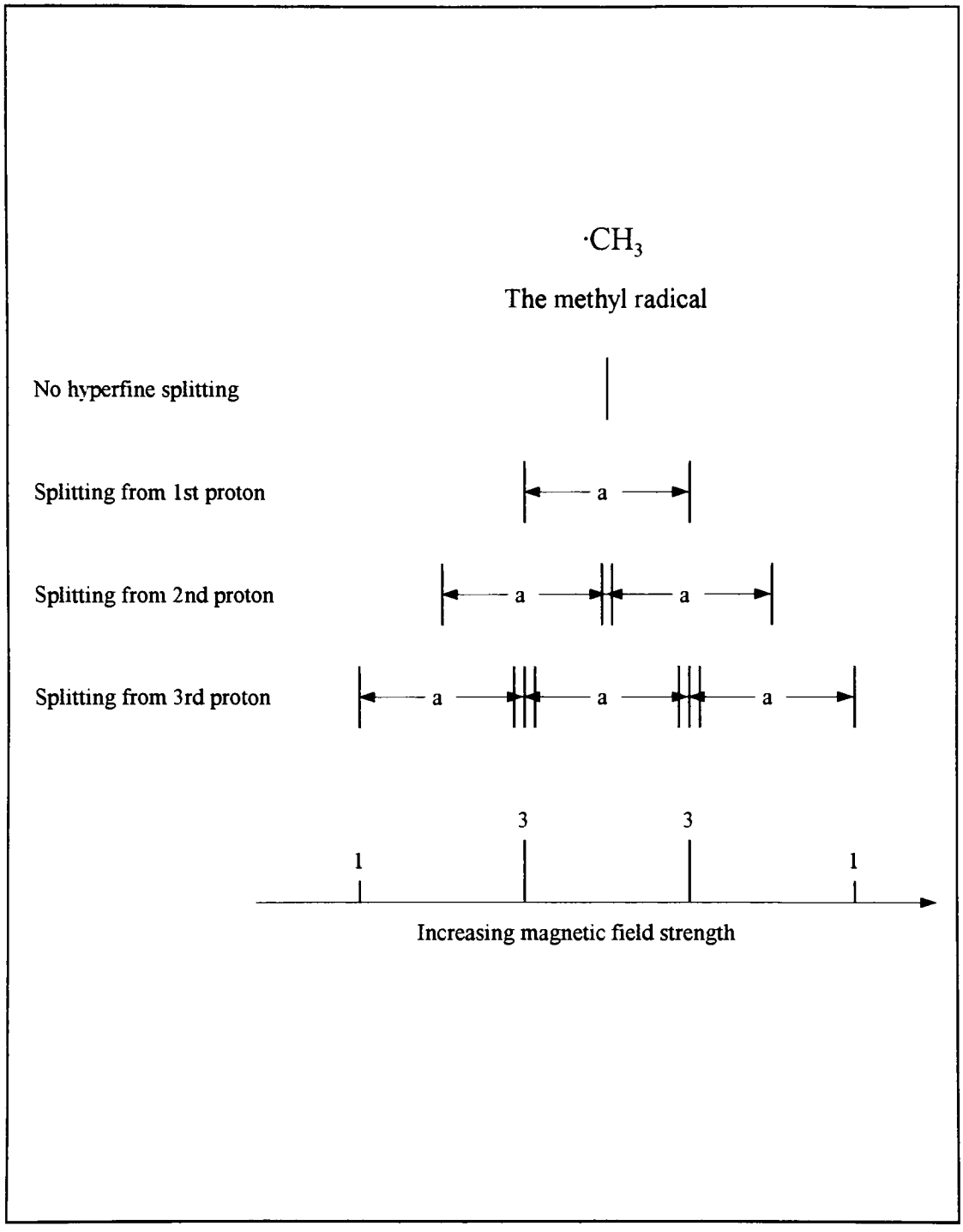


Figure 1-4 Hyperfine splitting of a spectral line, resulting from n-equivalent nuclei. Shown here is the methyl radical, having 3 equivalent protons and 1 unpaired electron. There are n + 1 = 4 hyperfine peaks. The degeneracy of spin energy states results in 1:3:3:1 relative peak intensities.

In general a single spin = I nucleus will split a spectrum into 2(I) + 1 hyperfine lines of equal intensity. Hydrogen has a spin = 1/2 nucleus, and splits a spectrum into 2 lines of equal intensity. Nitrogen has a spin = 1 nucleus, and splits a spectrum into 3 lines of equal intensity.

Although Equation 1-2 indicates that the condition of ESR absorption will occur at discrete values of field strength, B, actual ESR absorption lines have finite width, due to various homogenous and inhomogeneous line-broadening processes.

Inhomogeneous line broadening occurs mostly as a result of slight variations in the effective strength of the applied magnetic field throughout the sample region. Not all spins in the sample will be subjected to the same strength magnetic field. Spins experiencing the same field strength are considered as spin packets. Each spin packet will reach resonance at a slightly shifted value of the applied average magnetic field strength. Therefore, the resulting spectral line will actually be composed of the superposition of several closely spaced components that represent the resonance of different spin packets. The resulting envelope of spin-packets will generally be Gaussian in shape.

Homogenous line broadening occurs when the static or average magnetic field at each dipole is the same, but the instantaneous magnetic field is not. This occurs as a result of the interaction of the radicals with each other as well as their environment.

The interactions between the radicals and the environment, are often spin-lattice interactions. Spin-lattice interactions can occur between the spins of the radicals in the sample and the randomly oriented fields generated by surrounding lattice molecules. For example, if the radicals exist in water, the random magnetic fields generated from the protons of tumbling water molecules will superimpose with the applied magnetic field experienced by the radicals, as illustrated in Figure 1-5. The radicals can also experience spin-spin interactions between each other, such as electron-electron spin interactions, as well as electron-nuclear spin interactions. All of these interactions can lead to line broadening, to give rise to the characteristic Gaussian or Lorentzian line shapes of a typical ESR spectrum.

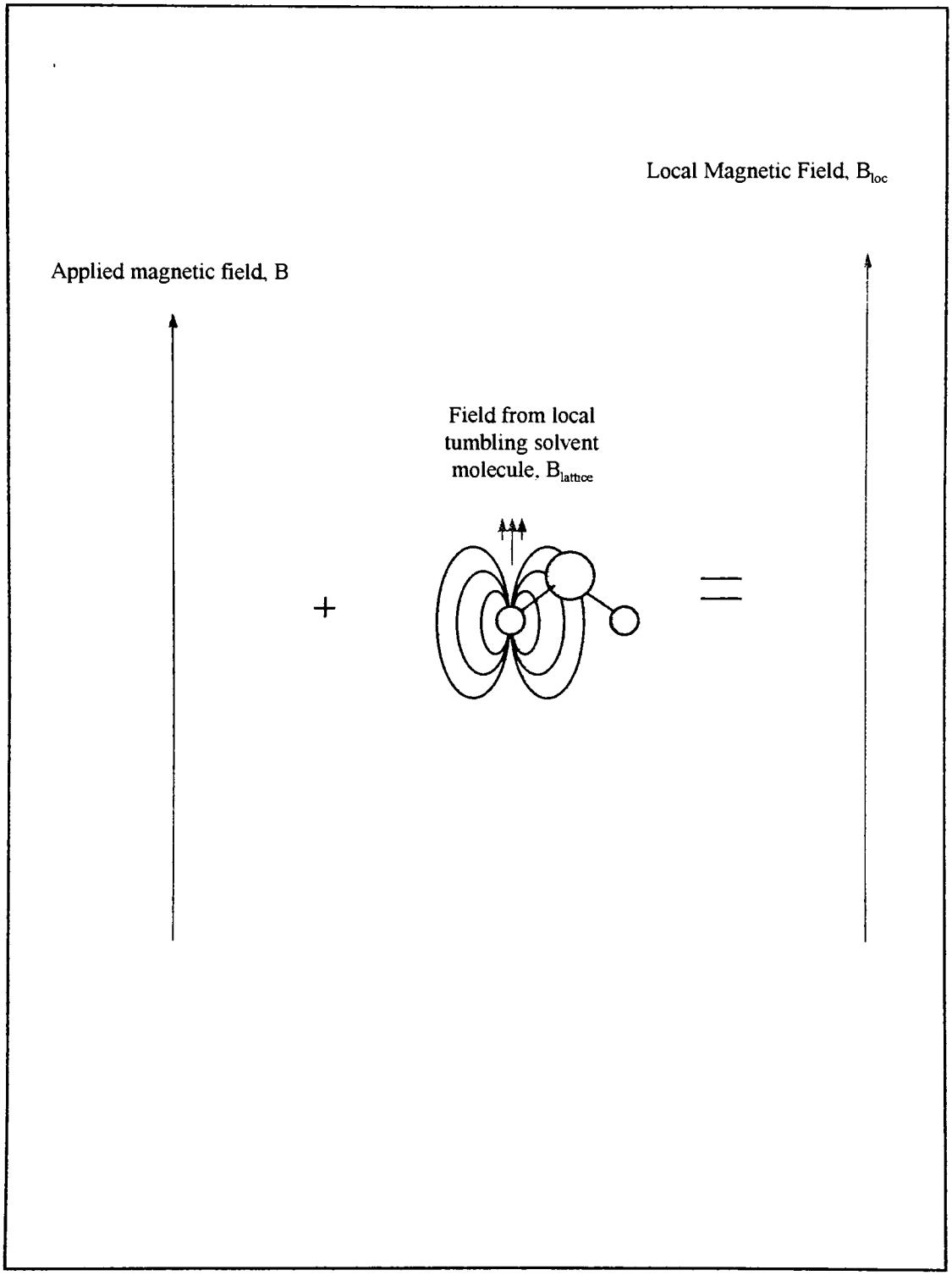


Figure 1-5 Spin-lattice interactions. The oscillating magnetic field that is generated by a tumbling molecule of the lattice will add vectorially to the applied magnetic field, causing local variations in the effective magnetic field.

1.2.2 *The LFESR Spectrometer*

The LFESR spectrometer can be considered as the integration of five principle subsystems, as illustrated in Figure 1-6. The subsystems are the: (1) Computer Interface; (2) RF Source; (3) RF Bridge; (4) Magnet, and; (5) Modulation / Detection Subsystem. The Computer Interface Subsystem is central to the four other subsystems, and consists of the hardware and software components that facilitate integrated computer control of the LFESR Spectrometer. Each subsystem consists of several components, as illustrated in Figure 1-7. The following sections describe the overall function of the LFESR spectrometer, with respect to the various subsystems.

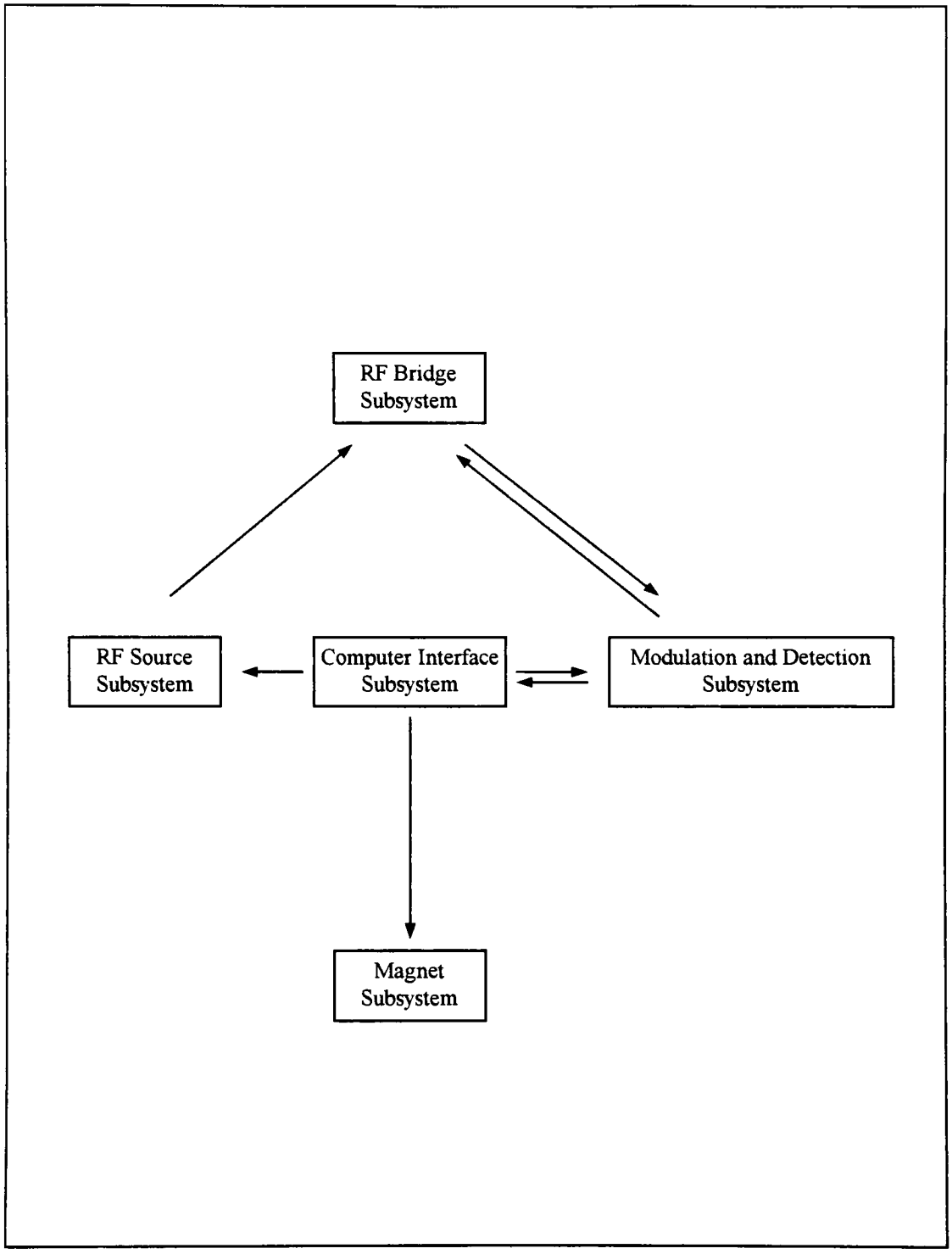


Figure 1-6 The five principle LFESR subsystems. This figure illustrates the central function of the Computer Control Subsystem, which enables programmable, integrated control of the spectrometer.

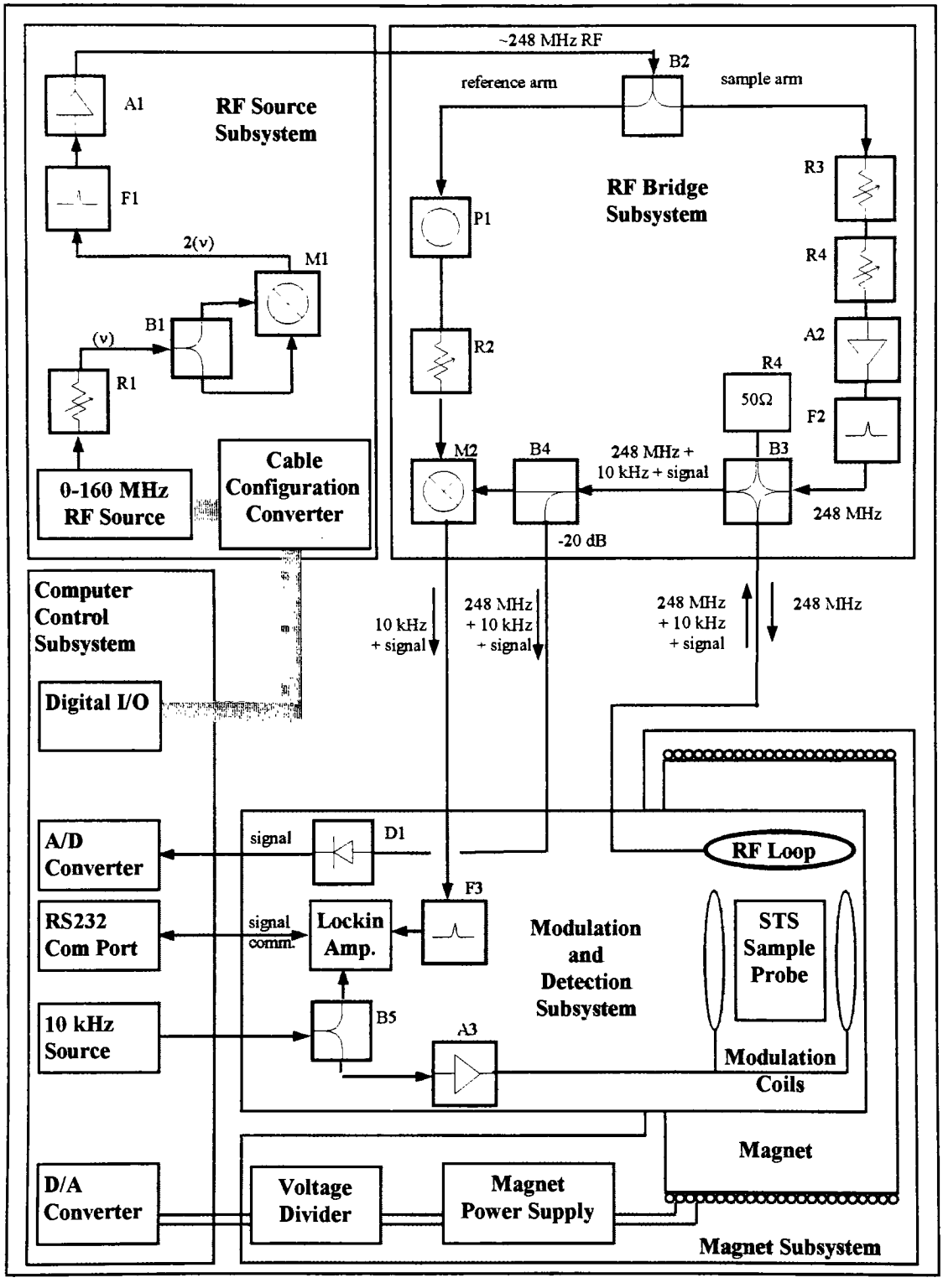


Figure 1-7 A block diagram of the LFESR spectrometer. Each of the five principle subsystems consist of several components as illustrated.

The RF Source Subsystem.

The fundamental component of the RF Source subsystem is a binary coded decimal (BCD) controlled, 0 to 160 MHz (±0.1 Hz) precision frequency generator, RF1, (PTS-160, Programmed Test Sources, Littleton, MA). The Computer Interface subsystem has been configured to control the frequency of RF1, through a digital interface, utilizing three 8-bit digital I/O ports (See "Interface Development" for details). The output of the PTS-160 is attenuated by a 0-10 dB (1 dB step) turret attenuator, R1, (5010A, Wavetek Corporation, San Diego, CA, USA). The attenuated RF power is then split by a 2-500 MHz power divider, B1, (PE 2000, Pasternack Enterprises, Irvine, CA, USA). Both sides of the power divider, B1, are directed into a 1-500 MHz doubly balanced mixer, M1, (DM-2A-200, Merrimac Industries, West Caldwell, NJ, USA), which functions in this configuration, as a frequency doubler. The frequency-doubled RF from the doubly balanced mixer, M1, is filtered by a 248 MHz LC bandpass filter, F1, and then preamplified by a 5-500 MHz, 22.5 dB gain signal amplifier, A1, (UTC-517-1, Aventek, Mitpitas, CA, USA).

The RF Bridge Subsystem.

The preamplified RF from A1 of the RF Source subsystem, enters the RF Bridge subsystem through a 2-500 MHz RF power divider, B2, (PE 2000, Pasternack Enterprises, Irvine, CA, USA). The power divider, B2, splits and directs the RF

power into the reference and sample arms of the RF Bridge subsystem. In the reference arm, the RF is phase-shifted with an adjustable length of RG-58 cable, P1, to be 90 degrees out of phase with the RF entering the 2-500 MHz doubly balanced mixer, M2, (DM-2A-200, Merrimac Industries, West Caldwell, NJ, USA) from the sample arm. The phase-shifted reference RF is attenuated by a 0-10 x 1 dB turret attenuator, R2, (5010A, Wavetek Corporation, San Diego, CA) before entering M2. In the sample arm, the RF is attenuated by a series pair of 0-70 dB (10 dB step) and 0-10 dB (1 dB step) turret attenuators, R3 and R4 respectively, (5010A and 5070A, Wavetek Corporation, San Diego, CA). The attenuated RF is amplified by a 150 kHz - 300 MHz, 10 watt power amplifier, A2, (411-LA, ENI Inc., Rochester, NY, USA), and filtered by a 248 MHz, LC bandpass filter, F2, before entering a 180 degree hybrid tee, return loss bridge, B3, (HJ-300, Merrimac Industries, Inc., West Caldwell, NJ, USA). The output B3 will consist of the sample arm RF, and the modulated voltage that results from any impedance mismatch between the 50 ohm reference load of B3 and the sample probe. Relatively large impedance mismatches result when the RF frequency is not tuned to the resonant frequency of the probe. Relatively small impedance mismatches result from ESR absorption of RF by the sample. A -20 dB portion of the output of B3 is directed to a 1 GHz diode detector, D1, (PE-8000-50, Pasternack Enterprises, Irvine, CA, USA), by a -20 dB directional coupler, B4, (CP-20-215, Merrimac Industries, Inc., West Caldwell, NJ, USA), and used for automatic frequency tuning. The remaining portion of the output of B3 is mixed with the phase

shifted reference RF at the doubly balanced mixer, M2, which acts as a detector of the modulated signal that results from the 180 degree hybrid tee.

The Modulation and Detection Subsystem.

At the heart of the Modulation and Detection subsystem is a single-turn-solenoid (STS) sample probe. The STS is a resonant LC circuit, which has been fabricated as part of this project. The details of the design and fabrication are given in the "Methods" section. The STS contains the sample (~15 mL), and is centered in a probe assembly which is oriented in the center of the spectrometer Magnet. The STS is inductively coupled to the source RF by an 18 gauge copper wire coupling loop. The coupling loop and STS are surrounded by a silver-plated brass shield. When the frequency of the source RF in the coupling loop matches the resonant frequency of the STS, an energetically dense B_1 field is produced in the region of the sample, and the probe is said to be in a tuned state. Field modulation is facilitated by a pair of saddle shaped coils which are placed on opposite sides of the STS, in a coaxial position with respect to the spectrometer magnet. The Computer Interface subsystem has been configured to supply low power 10 kHz RF to the Modulation and Detection subsystem, from a DC to 8 MHz frequency synthesizer, RF2, (Multigen Frequency Synthesizer, Sciteq Electronics, Inc., San Diego, CA, USA) of the Computer Interface subsystem (See "Interface Development" for details). The Low power RF is split by a 2-500 MHz power divider, B5. Part of the RF power is directed toward an audio

frequency amplifier, A3, (M-0.5t, Carver, Lynnwood, WA, USA) and used for energizing the field modulation coils. The remaining portion of RF power from B5 is used as reference at the Lockin Amplifier, LA1, (SR-530, Stanford Research Systems, Inc., Sunnyvale, CA, USA).

The modulated signal that is detected at the doubly balanced mixer, M2, of the RF Bridge subsystem, is filtered by a 10 kHz bandpass filter and detected again at the Lockin Amplifier, LA1, of the Modulation and Detection subsystem. The Computer Interface subsystem has been configured to poll the demodulated ESR signal data from the Lockin Amplifier through an RS232 interface (See "Interface Development" for details).

The -20 dB portion of power that is diverted by the directional coupler, B4, of the RF Bridge subsystem, is amplified by a 22.5 dB gain signal amplifier, A3, (UTC-517-1, Aventek, Mitpitas, CA, USA) and detected by a 1 GHz diode detector, D1, (PE-8000-50, Pasternack Enterprises, Irvine, CA, USA). The Computer Interface subsystem has been configured to digitize the rectified signal from the diode detector, D1, through an analog-to-digital (A/D) interface, and to utilize the digitized data in automatic frequency tuning functions of the spectrometer (See "Interface Development" for details).

The Magnet Subsystem.

The central component of the Magnet subsystem is the magnet. The magnet is an 800 AT^{-1}, 30 cm diameter x 45 cm long solenoid, consisting of 800 turns of 10 gauge HOK copper wire. At 8.5 mT, the field produced by the magnet is constant to within ±0.015 mT over the region of the STS sample probe [Szczepaniak, 1993]. The magnet is energized by a 0-30 A, DC current regulated power supply, PS1, (3002-1, Systron Donner, Alpha Subsidiary, Oakland, CA, USA). The Computer Interface subsystem has been configured to control the current output of the magnet power supply, PS1, through a digital-to-analog (D/A) interface (See "Interface Development" for details).

A complete LFESR component list is given in Appendix A. This list includes components shown in Figure 1-7 Figure 1-7 reflects the state at which the spectrometer exists at the completion of the development of the Computer Interface Subsystem.

1.3 Interface Development

The Computer Interface Subsystem was developed in several stages, corresponding to the discrete tasks that were to be facilitated by the interface, as well as the functional hierarchy of each task, as illustrated in Figure 1-8. The various hardware elements of the interface are detailed in the following sections, while corresponding software development is detailed in Appendix F

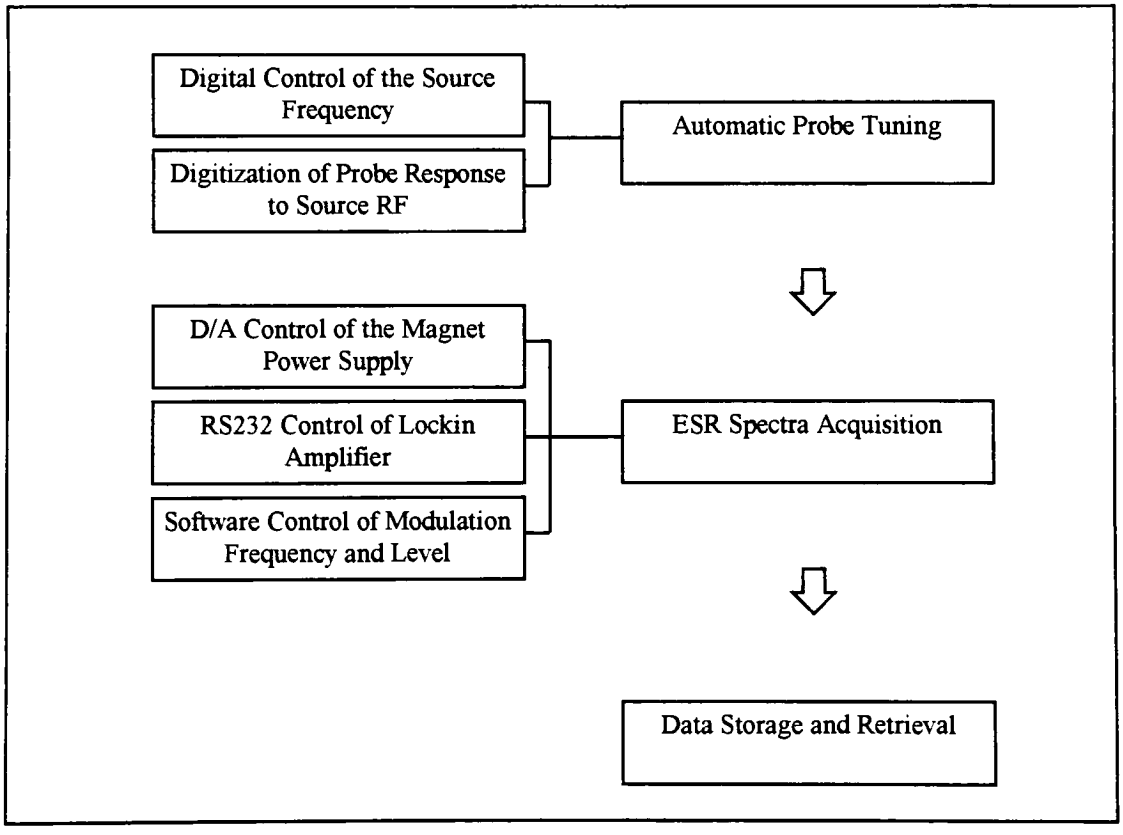


Figure 1-8 A block diagram illustrating the principle stages and flow of development of the Computer Interface Subsystem.

1.3.1 *Computer Hardware Configuration*

A 486 DX2-66 computer (Zeos International Limited, St. Paul, MN, USA) serves as the development platform for the Computer Interface subsystem. The computer is equipped with a 1 MHz digital and analog interface board (Win-30, Omega Engineering Inc., Stanford, CT, USA), a DC to 8 MHz frequency synthesizer, RF2, (Multigen Frequency Synthesizer, Sciteq Electronics, Inc., San Diego, CA, USA), and a standard RS232 serial communications port.

A 16-bit, 0 to 5 volt digital-to-analog (D/A) converter, of the Win-30 is used to control a current regulated power supply, PS1, (Model 3002-1, Systron Donner, Alpha Subsidiary, Oakland, Ca) of the Magnet subsystem. A 12-bit, 0 to 5 volt analog-to-digital (A/D) converter is used to digitize the probe-tuning signal that results at the 1 GHz diode detector, D1, (PE-8000-50, Pasternack Enterprises, Irvine, CA, USA), of the Modulation and Detection subsystem. Three 8-bit digital I/O ports of the Win-30 are used to facilitate binary coded decimal (BCD) control of a 0 to 160 MHz, ±0.1 Hz precision frequency synthesizer, RF1, (PTS-160, Programmed Test Sources, Littleton, Ma., USA) of the RF Source subsystem. The RS232 serial communications port is used to communicate with the lockin amplifier, LA1, (SR-530, Stanford Research Systems, Inc., Sunnyvale, CA, USA) of the Modulation and Detection subsystem.

1.3.2 *Digital Control of the RF Source Frequency*

In order to achieve computer control of the RF source frequency, a digital interface was implemented between the three 8-bit digital I/O ports of the Win-30, and the program connector of the PTS-160 frequency generator, RF1. The PTS-160 program connector allowed the on-board digit control modules of the PTS-160 to be programmed by parallel BCD input from the Win-30 digital I/O ports. Each digit control module of the PTS-160 is programmed by 4-bit parallel BCD input. The input to each control module is controlled by logic gates. If a gate latch is set high, the input is not stored by the module. If a gate latch is set low, the input is stored by the module, and the digit value is updated accordingly. The four control bits of each digit module correspond to four pins of the program connector. The value of a digit is programmed by applying the 4-bit BCD equivalent of the desired digit value, to the four corresponding pins of the digit module (See Figure 1-9). The digit value is updated by setting the corresponding gate latch low. The remote operation of the PTS-160

Digit (MHz)	BCD Weight:				Latch Enable
	1	2	4	8	
	Pin Numbers				
10	15	16	40	41	23
1	17	18	19	20	24
0.1	1	2	26	27	24
0.01	3	4	28	29	25
0.001	5	6	30	31	25

Figure 1-9 The pin numbers and BCD weights for selected digit control modules, and the latch enable functions of the PTS-160 are listed. Remote Enable = Pin 42.

through the program connector can be enabled or disabled by setting the remote-enable bit of the program connector (pin 42), high or low, respectively. All of the program connector functions are negative-true (i.e. applying 0 volts at a pin is equivalent to setting the corresponding bit high).

Five digit modules will be programmed through the Computer Interface. The digit modules correspond to the 10, 1, 0.1, 0.001 and 0.0001 MHz digits of the PTS-160. The 10 MHz digit module can be programmed to have values from 0 to 14, while the remaining digit modules can be programmed to have values from 0 to 9. This arrangement allows 6-digit precision, with five digit modules, and requires 20 bits of BCD input (5 digits x 4 bits). The five digit modules are controlled by three gate latches, corresponding to three pins as indicated in Figure 1-9, and require 3 separate bits of BCD input. Finally, the single remote enable pin of the program connector requires 1 bit of input. This means that a total of 24 bits of input were required to perform the interface. The three 8-bit digital I/O ports of the Win-30 were used to supply the required 24 bits of input to the PTS-160 program connector.

Although the three 8-bit ports of the Win-30 were sufficient to supply the 24 bits of required input to the PTS-160 program connector, the design of the Win-30 only allows BCD data to be transmitted in byte (8-bit) sized pieces. This means that individual pins of the Win-30 cannot be *conveniently* correlated to the appropriate pins of the PTS-160 program connector, within the software. This problem was addressed

by designing and fabricating a custom ribbon-cable interface board, that would appropriately correlate the pins of the two systems (See Figure 1-10). The cable interface board was fabricated from a pair of dual-row 50-pin female header strips, interconnected by hard-wire solder connections, to result in the desired pin-to-pin correlations.

Since each digit module required only 4-bits of information, each byte of BCD input from the Win-30 had enough information to program two digit modules. Five half bytes (nibbles) were required to program the five separate digit modules. The low and high nibbles of Port A were used to transmit the BCD input for the 0.001 and 0.01 MHz digit modules, respectively. The low and high nibbles of Port B were used to transmit the BCD input for the 0.1 and 1 MHz digit modules, respectively. The low nibble of Port C was used to transmit the BCD input for the 10 MHz digit module, while the high nibble of Port C was used to transmit the BCD input for the three separate gate latches, as well as the remote enable. The operation of the Win-30 allows for the the bit-wise state of a single given 8-bit digital I/O port to be altered at a time. At first, this created unwanted transient frequency values to occur, since not all digits are being altered simultaneously. This problem was solved by appropriately programming the gate latches, to ensure that the values of all five digit modules were updated simultaneously. For software details, see Appendix F.

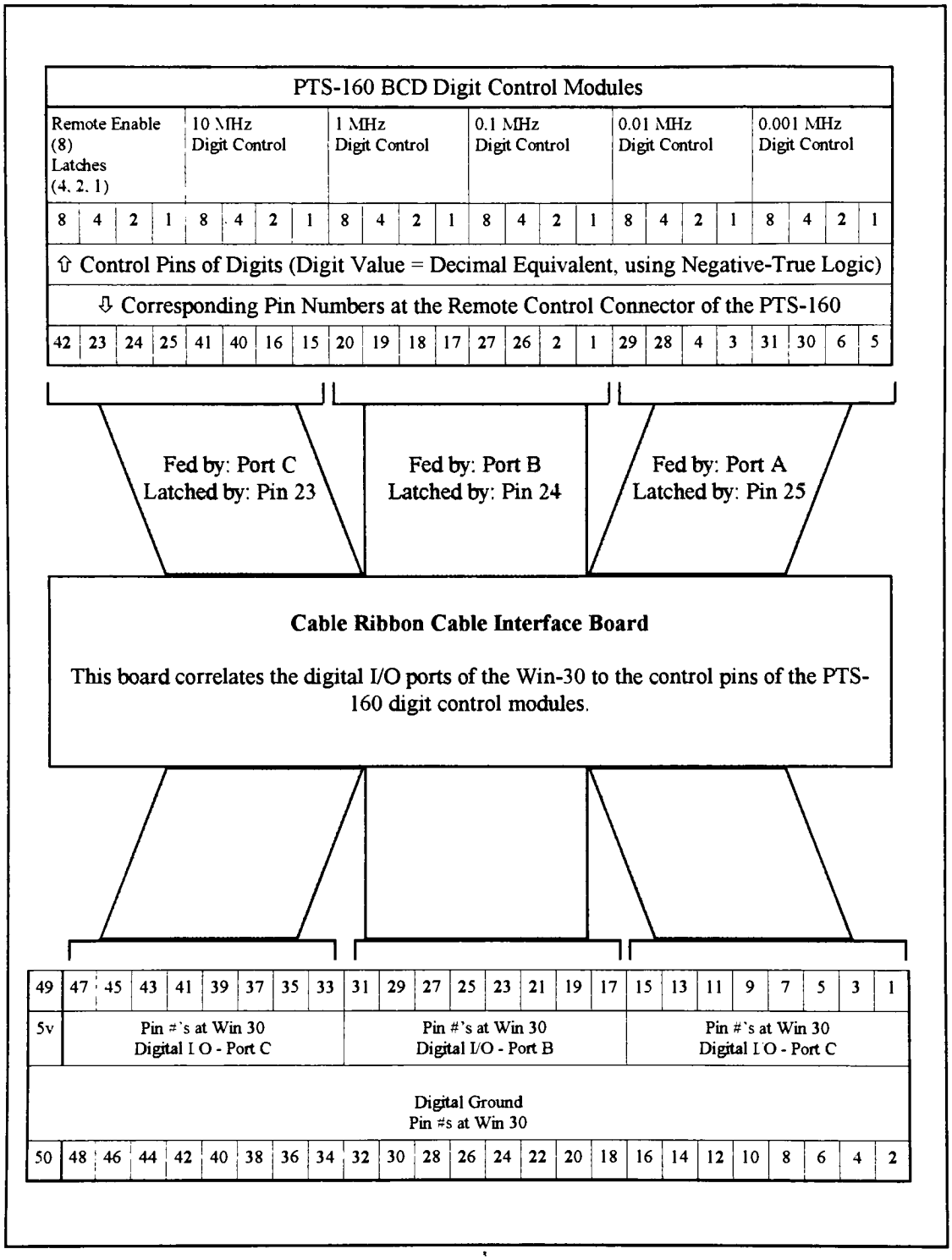


Figure 1-10 A schematic illustration, showing the function of the Custom Ribbon Cable Interface Board, that was fabricated to facilitate digital control of the RF source frequency.

1.3.3 *Digitization of the STS Frequency Response.*

The rectified voltage that results at the diode detector, D1, of the Modulation and Detection subsystem, is a measure of the amount of RF power that is reflected by the STS sample probe. Probe tuning involves the minimization of the RF power reflected by the probe, and therefore requires the ability of the operating software to interpret the voltage at D1. This has been accomplished by implementing an analog-to-digital interface between the output of the detector diode, D1, and the 0-5 volt dc, 12-bit A/D detector of the Win-30 interface board. The output of D1 was connected at pins 1 and 2 of the Win-30, for differential input across channels 1 and 8, respectively. The signal input was referenced to ground with a 1 kilohm resistor between the ground side input (channel 0, pin 1), and the analog ground of the Win-30 (pin 3). The interconnections are shown in Figure 1-11. For software details, see Flow Chart F-2 and Program Block F-3.

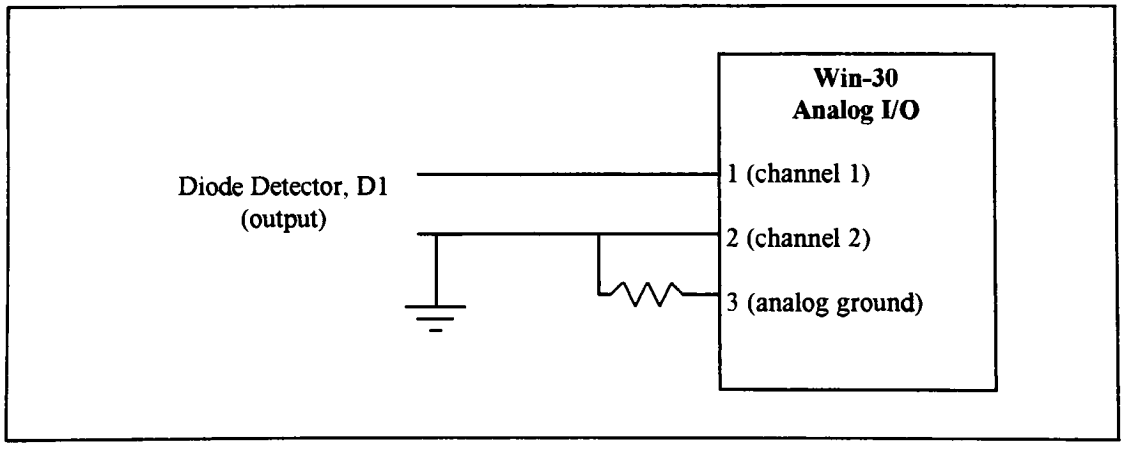


Figure 1-11 The interconnections between the output of the detector diode, D1, and the Win-30.

1.3.4 *Automatic Probe Tuning.*

Automatic probe tuning was accomplished by combining the functions of the digital interface with the PTS-160, the A/D interface with the detector diode, D1. Software was written to perform frequency sweeps, according to user-defined sweep parameters of Center Frequency, Frequency Width, and Frequency Step. The program was written to provide the system operator with a real-time graphic display of the probe-tuning sweeps, that is displayed within a daughter window of the LFESR system operating window. This allows the operator to view the results of inductive coupling adjustments made during the probe tuning sweeps. The program was written to perform as many sweeps as the operator specifies, and to analyze for the resonant frequency of the STS for each sweep. After a set of probe-tuning sweeps has been made, the PTS-160 is set and locked at the resonant frequency determined by the last sweep. The program was also written to save the probe-tuning A/D data, along with probe tuning parameters to a user-defined disk file, making the data available to external data processing software, such as common Windows-based spread sheet programs. For software details, see Appendix F.

1.3.5 *D A Control of the Magnet Power Supply.*

The magnet is powered by a current regulated power supply (Model 3002-1, Systron Donner, Alpha Subsidiary, Oakland, Ca). The power supply is capable of delivering 0 to 30 amps dc. The output of the power supply is governed by a 0 to 1v dc control voltage supplied to pins 9 and 10, located on the back panel of the instrument. With an optimum load impedance, a control voltage of 1v would produce a maximum output of 30 amps to the magnet. However, the maximum output of the power supply is internally limited based on the load impedance of the magnet being used, to have an upper limit of 18 amps. This means that the full working range of the power supply (0 to 18 amps) is achieved by using a control voltage of 0 to 0.6v dc.

A 16-bit -5v to +5v digital-to-analog (D/A) converter of a digital and analog interface board was used in conjunction with a custom voltage divider to provide the 0 to 0.6v dc control voltage to the power supply's control pins. By using the voltage divider, the resolution contained in the 0 to +5v working range of the D/A converter was compressed into the much smaller 0 to 0.5v control voltage range of the power supply. Since the D/A converter has 16-bit resolution across -5v to +5v, the 10v range can be divided into 2^{16} (= 65536) steps. The positive 0 to 5v range of the D/A converter contains 65536 / 2 (= 32768) steps. Using the voltage divider results in a control voltage resolution of 0.5v / 32768 steps (= 0.000015v / step). See Figure 1-12.

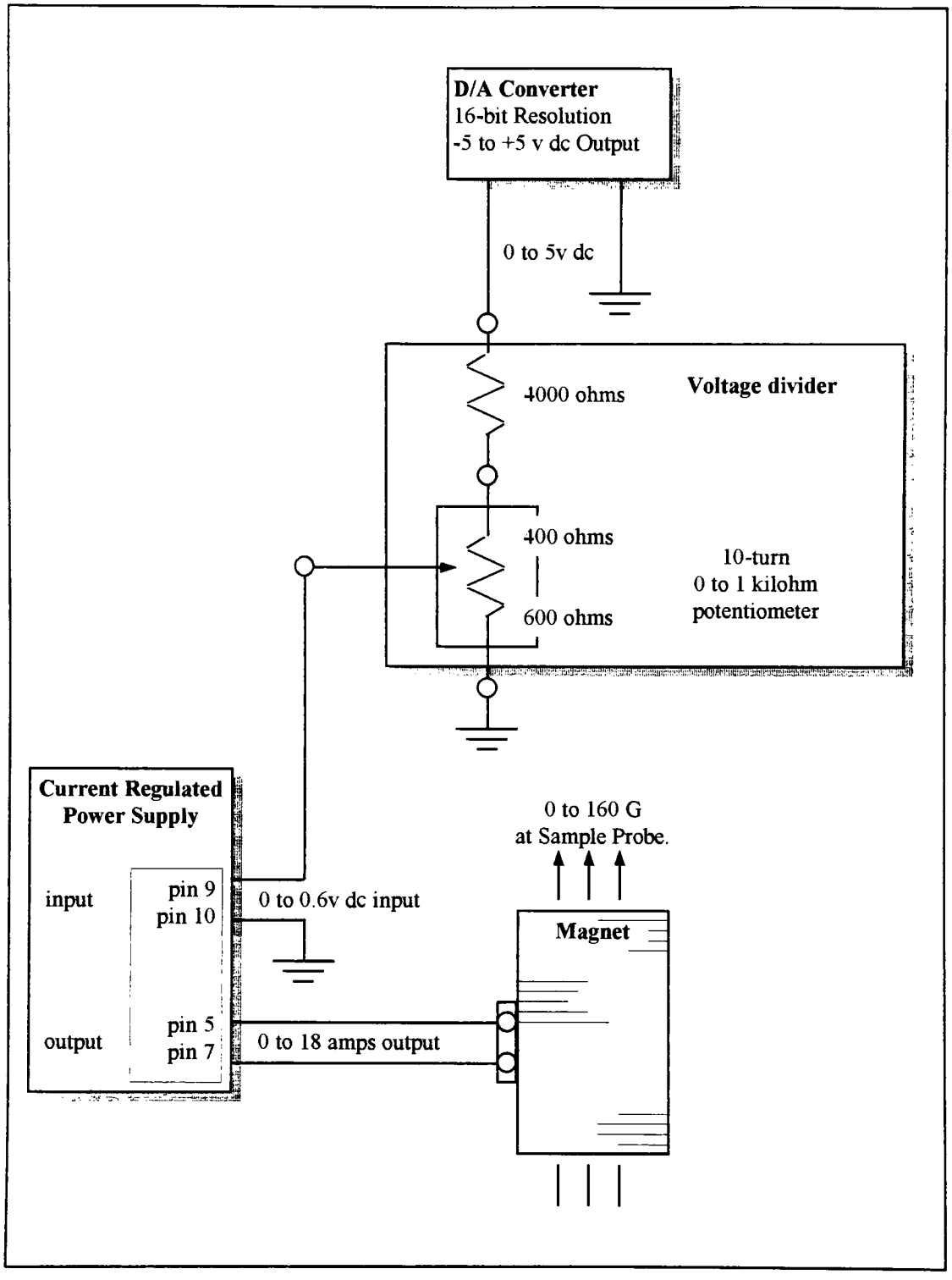


Figure 1-12 A schematic representation of the D/A control of the Magnet Subsystem.

1.3.6 *RS232 Control of the Lockin Amplifier*

Control of the Lockin amplifier was accomplished by an RS232 serial interface with the computer. An RS232 cable was directly connected between the serial port of the computer, and the RS232 interface connector on the back panel of the Lockin amplifier. For software details, see Appendix F.

1.3.7 *Software Control of the Field Modulation Frequency and Level.*

Control of the modulation frequency and level was accomplished by installing a 0-8 MHz frequency synthesizer (Multigen Frequency Synthesizer, Sciteq Electronics, Inc., San Diego, CA, USA) on the back plane of the computer, and interconnecting the output of the frequency synthesizer to the power divider, B5, of the Modulation and Detection subsystem. Multigen drivers were placed under the "Set-Multigen" start-up icon in the LFESR program group. This enabled the parameters of the Multigen frequency synthesizer to be varied, simply by opening the drivers through the start-up icon, and entering the values for the parameters to be set.

1.3.8 *ESR Spectra Acquisition.*

ESR spectra acquisition was accomplished by combining the various interface functions that have been developed, through integrated software development. For software details, see Appendix F

1.3.9 *Calibrating the D/A control of the Magnet Subsystem.*

The Sweep Parameters and the associated software algorithms were calibrated based on the g-factor and hyperfine splitting constant of the 2,5-di-tert-butyl parabenzosemiquinone anion radical (g = 2.004, a = .21 mT). The spectrum has three hyperfine components, each separated by 0.21 mT. Based on the g-factor of 2.004 and a Larmore frequency of 248 MHz, the center of the spectrum should be located at 8.8 mT. The radical was prepared as described in Figure 1-13. The sample tube was filled to the top with the radical solution, and placed in the probe. The sample probe was auto-tuned to a resonance frequency of 246760 kHz.

A preliminary version of the operating software was written to allow a scan to be performed, in which the D/A converter was fed digital input values which ranged from 32767 to 65535 in steps of 32. This resulted in the D/A converter output to be 0 to 5v dc. The 0 to 5v output of the D/A converter was scaled at the voltage divider to be 0 to 0.6v. This voltage was applied to pins 9 and 10 of the magnet power supply. This resulted in the capability to perform a full-range-sweep of the magnet.

A spectrum of the radical was recorded over the full-range-sweep of the magnet (See Figure 1-14a). The data contained in the spectrum was closely examined in the region of the spectral peaks (See Figure 1-14b), to relate units of digital input at the D/A converter to units of magnetic field strength at the sample. This information was utilized in the development of a refined version of the software, that contained

algorithms to allow the user to input sweep parameters in units of mT.

The Sweep calibration data in Figure 1-14b shows that a D/A input of 52280 digital units corresponds to a magnetic field strength of 8.8 mT. In addition, the spectrum reveals that a sweep of 420 digital input units at the D/A converter results in a sweep of 0.21 mT at the sample. These results have been fit to a linear relation between digital input units at the D/A converter and magnetic field strength units at the sample.

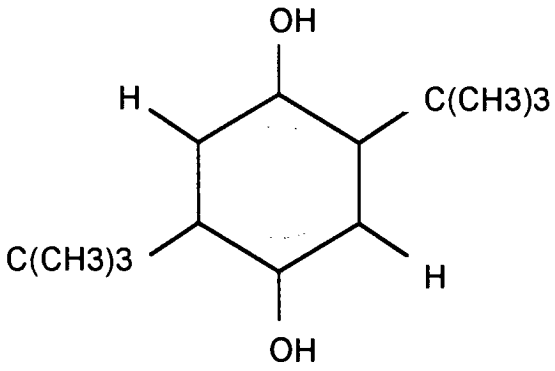
$y = mx + b$
52280 digital units = (2000 digital units/ mT)(8.8 mT) + b
b = 34680 units

Therefore,

y digital units = (2000 digital units/mT)(x mT) + 34680 digital units

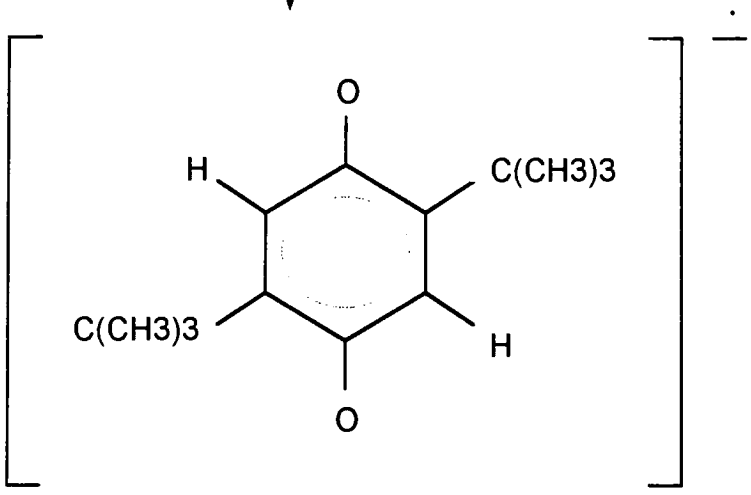
This is the result which was used to hard code the software, in order to allow for the input of sweep parameters in units of mT.

Once the sweep controls were calibrated, the system was tested with respect to the sweep parameters. Tests were carried out on the 2,5-di-tert-butyl parabenzosemiquinone anion radical, which was prepared for the calibration discussed above. As a performance test, the radical was scanned using Sweep Start value of 8.4 mT, a Sweep Width value of 0.8 mT, and a Sweep Resolution value of 0.0005 mT. The spectra is shown in Figure 1-15. The scan reflects sweep accuracy with respect to the sweep parameters.



2,5-di-tert-butyl hydroquinone

Ten pellets of KOH were dissolved in 20 mls of 50:50 ethanol and water. To the KOH solution, half of a micro-spatula of 2,5-di-tert-butylhydroquinone was added and mixed thoroughly. This produced an red-orange colored solution of the 2,5-di-tert-butyl parabenzosemiquinone anion radical.



2,5-di-tert-butyl parabenzosemiquinone anion radical

Figure 1-13 Preparation of the 2,5-di-tert-butyl parabenzosemiquinone anion radical from 2,5-di-tert-butylhydroquinone.

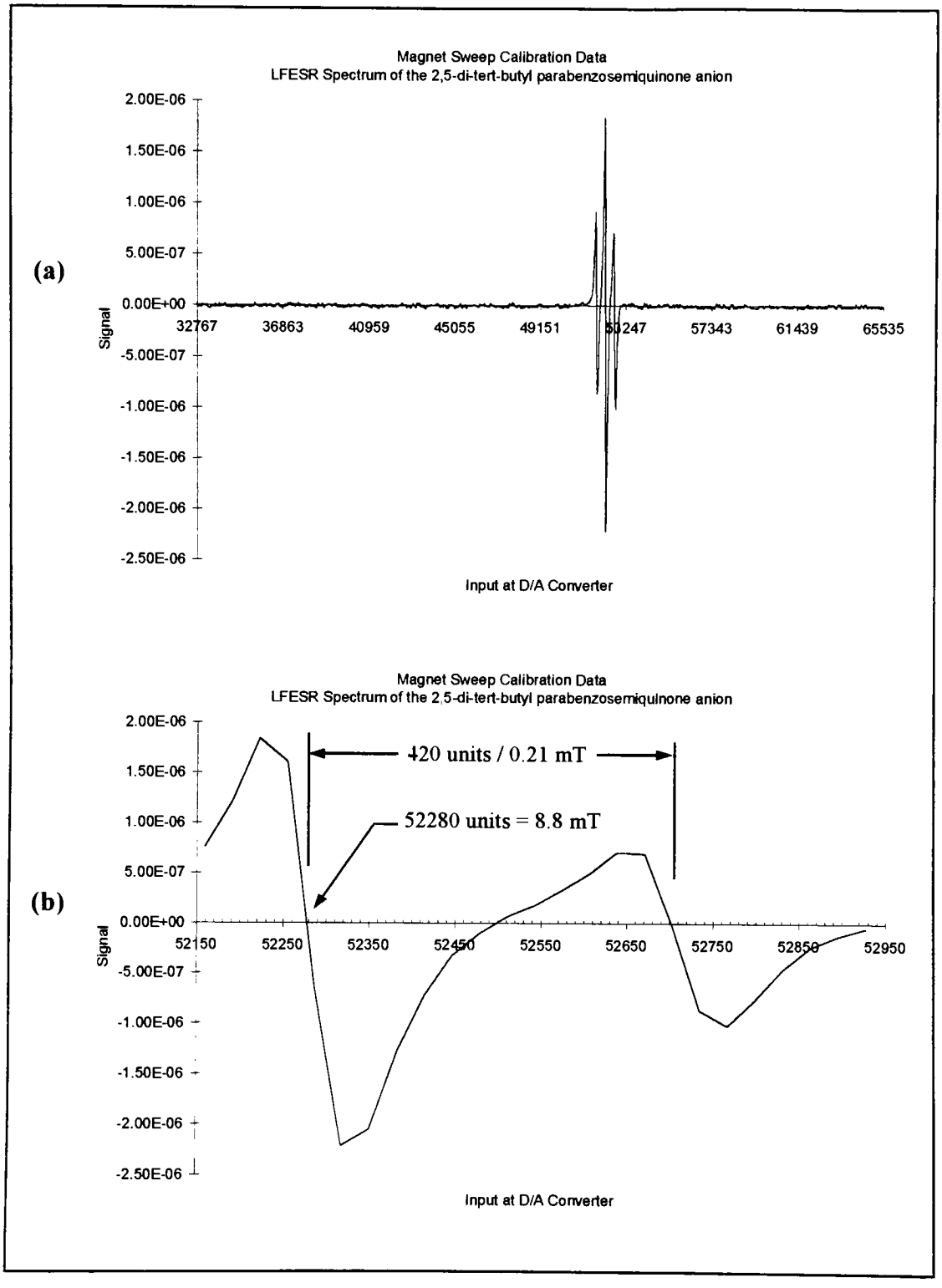


Figure 1-14 Magnetic Field Calibration data. **(a)** Full sweep of magnetic field strength range. **(b)** A close-up of the spectral peaks of the calibration scan.

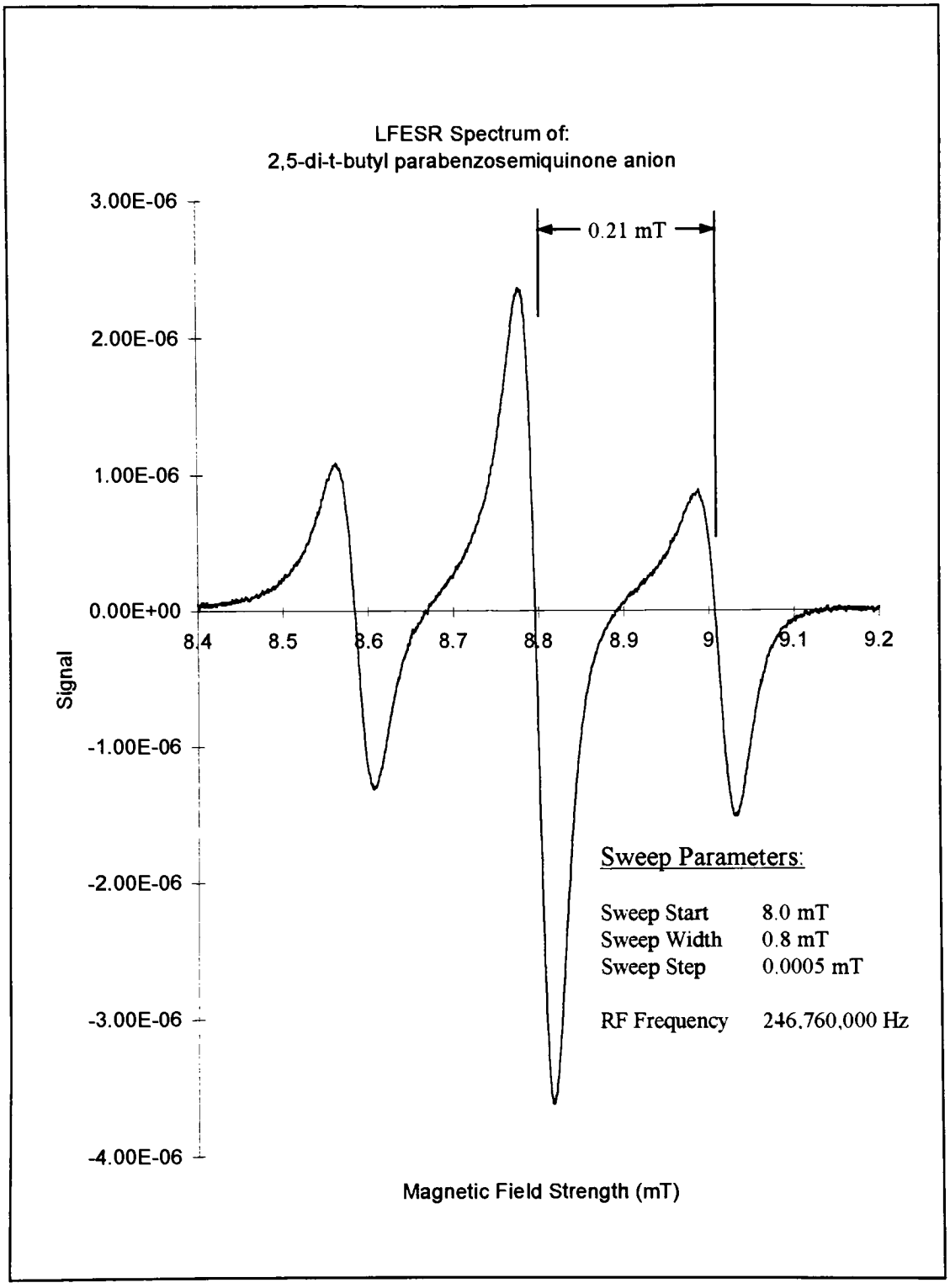


Figure 1-15 Test scan of 2,5-ditert-butyl parabenzosemiquinone anion, prepared as illustrated in Figure 1-19.

Part II

Performance Characterization of the LFESR Spectrometer

2.1 Introduction

The development of a new piece of analytical instrumentation generally requires that the performance of the instrument be carefully characterized, before it is to be used for quantitative measurements. Well designed and executed performance characterization tests reveal the effects of different operating factors on key measures of instrumental performance such as sensitivity and resolution. The goal here is to

characterize the performance of the LFESR spectrometer that was interfaced in Part I of this research project. The results of the performance characterization will be used to optimize the newly interfaced spectrometer.

Several parameters are expected to have an effect on the sensitivity and resolution of the LFESR spectrometer, including the RF frequency and power, the modulation frequency and amplitude, the temperature of the sample, the concentration of analyte in the sample, the Q factor of the sample probe, and the level of signal averaging that is used during spectra acquisition. In the current configuration, RF frequency, modulation frequency, temperature, and the Q-factor of the sample probe, are treated as constants, while modulation amplitude, RF power, concentration of analyte and the level of signal averaging are treated as variables. Modulation amplitude is controlled by varying the setting of field modulation power output at the field modulation power amplifier. RF power is controlled by varying the setting of RF power attenuation between the RF bridge and the STS sample probe.

The goal of characterizing the performance of the LFESR spectrometer will be met by carrying out a series of tests, designed to experimentally determine the specific effects of various system operating variables on the performance of the LFESR spectrometer. A good aqueous spin standard for the characterization of ESR spectrometers is potassium peroxylamine disulfonate [$K_2NO(SO_3)_2$], (Fremy's salt). This is a good standard for aqueous solutions, since concentrations can be determined

optically. Standard solutions are typically prepared in ten percent sodium carbonate. Solutions are quite stable and have shelf lives of months if refrigerated. While the concentration may be slightly less stable at room temperature, or at the slightly elevated temperature of the STS sample probe, the concentration stability can be checked by measuring the optical absorbance of the solution before and after experimental scans. The ESR spectrum of Fremy's salt consists of three equally intense lines separated by 1.31 mT (Wertz and Bolton, 1972). The UV-visible spectrum of Fremy's salt consists of absorption peaks at 248 nm and 545 nm, with corresponding molar absorbtivities of 1690 $M^{-1}cm^{-1}$ and 20.8 $M^{-1}cm^{-1}$, respectively. Optical measurements are typically made at the 250 nm wavelength (Jones, 1963).

First, a spectrum of Fremy's salt is taken and used to verify the expected relative peak intensities and hyperfine splitting constants. This test is further used to confirm that the length of the phase-adjusting loop, P1, of the RF bridge is properly adjusted to result in line symmetry. Second, the effects of the field modulation power setting on first derivative line shape are tested. Third, the effects of the RF power attenuation on first derivative line shape are tested. Fourth, the relationship between line intensity and spin concentration is determined. Fifth, the relationship between SNR and spin concentration is determined. Finally, the relationship between SNR and signal averaging is determined. The results of each of these tests are discussed with respect to optimization of sensitivity and resolution.

2.2 Background and Theory

The first derivative spectral lines obtained in LFESR scans, are defined here by peak-to-peak derivative amplitude, A_{pp}, and peak-to-peak derivative width, B_{pp}, as shown in Figure 2-1. The software was written to automatically analyze spectral data for the values of A_{pp} and B_{pp}, and automatically display the values of each on the computer screen at the completion of a scan. However, the automatic peak-to-peak analysis is based on searching for the absolute maximum and minimum in the spectral data, and does not allow for the possibilities that the absolute maximum and minimum may correspond to two different spectral lines of the scan, or to noise in the scan. The use of this feature requires prudence on the part of the experimenter. The most certain result of A_{pp} and B_{pp} is obtained by manually searching the serialized data for the values. This latter method will be used for the analysis of all performance characterization test data described in this document.

The signal-to-noise ratio (SNR) is defined here as the ratio of the peak-to-peak derivative amplitude and the standard deviation of the baseline response. The baseline response is considered here to be represented by any portion of the scan that is more than five times the peak-to-peak derivative width away from the nearest spectral line. In all cases, the standard deviation of the baseline response is based on the average of at least twenty consecutive data points.

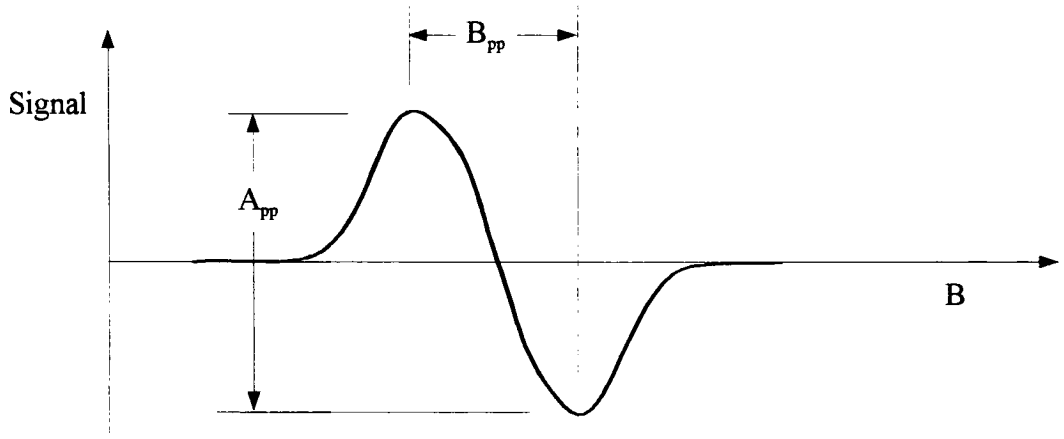


Figure 2-1 First derivative line parameters. Peak-to-peak derivative amplitude, A_{pp}, and peak-to-peak derivative width, B_{pp}.

Several factors effect sensitivity and resolution in ESR spectrometers, including modulation frequency and amplitude, RF frequency and power, temperature, spin concentration, and the level of signal averaging utilized in the acquisition of spectral data. In the current configuration of the LFESR spectrometer, modulation frequency, RF frequency and temperature are treated as system operating constants, while modulation amplitude, RF power and signal averaging are treated as system operating variables. The current design of the RF bridge circuitry, fixes the modulation frequency at 10 kHz. Although the RF frequency is variable from 0 to 160 MHz, the operating frequency is fixed at the resonant frequency of the STS sample probe, which is approximately 248 MHz in the current configuration. The temperature within the STS sample probe is considered a constant value, once the instrument has fully warmed up and reached equilibrium with the surroundings. The actual equilibrium temperature may vary slightly, depending on room temperature. However, the temperature for a critical series of scans is controlled by performing the scans under similar conditions of climate, including temperature as well as air flow through the magnet/sample probe assembly.

Field modulation amplitude should be a small fraction of the true peak-to-peak derivative widths for spectral lines of interest. In order to result in the accurate first derivative response, the portion of an absorption line scanned during one half cycle of field modulation should be nearly linear. As field modulation amplitude is increased,

the value of A_{pp} will first increase in a nearly linear relationship to increases in field modulation amplitude, eventually reaching a maximum, and then falling off as the modulation amplitude becomes excessive. The first inclination is to set the field modulation amplitude to a value that corresponds to the maximum value of A_{pp}. However, before the maximum A_{pp} is obtained, B_{pp} broadening will occur, resulting in a loss of line shape accuracy and resolution. Therefore, the optimum setting for modulation amplitude will depend on the balanced requirements of sensitivity and resolution (or accuracy of line shape), for a particular experiment (Wertz and Bolton, 1972).

In theory, the square of the peak-to-peak derivative amplitude, A_{pp}^2, will be proportional to the RF power, P, that is incident upon the sample (Wertz and Bolton, 1972). However, if the RF power level is excessive, then saturation will occur. At the onset of saturation, A_{pp}^2 will increase at a less than linear rate with respect to increases in P, and peak-to-peak derivative line width broadening may occur.

The sensitivity of the LFESR spectrometer is defined here as the slope of the calibration plot of intensity vs. absolute number of spins. Intensity is defined as the sum of the intensities of all lines in the spectrum. The intensity of each line in the spectrum is most accurately determined by calculating the double integral of the first derivative data, corresponding to the line. The double integration should be calculated over a sufficiently wide range of first derivative data. A sufficient scan range is defined

here as eight times the peak-to-peak derivative width. For concentrations less than 10^{-4} M, line intensity is expected to be directly proportional to the absolute number of spins corresponding to the line (Wertz and Bolton, 1972). At concentrations exceeding 10^{-4} M, electron spin-electron spin exchange interactions will lead to distortions away from true line shape and the loss of linearity in the relation of intensity vs. concentration.

If a plot is made of SNR vs. concentration of Fremy's salt, the concentration, and therefore total number of spins, required for an SNR value of 10 or greater can be determined. Dividing the total number of spins by three (there are three equally intense peaks in the Fremy's salt spectrum) results in the number of spins per peak. Dividing the number of spins per peak results in the line width-dependent sensitivity limit of the spectrometer, measured in units of number of spins / mT.

The technique of signal averaging may be used to improve the SNR of a spectrum. Generally, the relative improvement in SNR is directly proportional to the square root of the number of samples that are averaged for each data point represented in the spectrum. However, when dealing with short-lived species, the cost of added acquisition time associated with signal averaging, may outweigh the benefit of improved SNR.

2.3 Experimental Methods

Each of the following procedures will utilize aqueous solutions of potassium peroxylamine disulfonate, $K_2NO(SO_3)_2$ (Fremy's Salt), prepared in 10% sodium carbonate in HPLC water. Optical measurements of solution concentration were made before and after experiments, to confirm concentration stability within the time frame of the experiment. Optical measurements were made at 248 nm, using a UV-Vis Spectrometer (Lambda 4A, Perkin-Elmer Corporation, Norwalk, CT, USA). All experiments were carried out at the ambient operating temperature of the spectrometer sample probe (~27 °C). The temperatures of sample solutions were allowed to equilibrate with the temperature of the probe, before probe-tuning operations and LFESR scans were performed. All LFESR sample volumes are 15 mL.

Note: Unless otherwise indicated, all scans will be performed with the RF power attenuation set at 10 dB, and the field modulation power level set at 1 watt / 8 ohms, as indicated by the respective front panel instrumental controls. See Appendix B for specific instructions on operating the LFESR spectrometer. Note also that the software default value of n = 1 will be used for signal averaging (i.e. no signal averaging will be used), unless otherwise indicated for a given scan or set of scans.

2.3.1 *LFESR spectrum of Fremy's salt.*

An LFESR scan was made of a 75 μM solution of Fremy's salt by sweeping a range of 7.000 to 10.800 mT, in steps of 0.005 mT (See Results, Figure 2-2). The scan data was used to verify the expected hyperfine splitting and relative peak intensities of the three line spectrum of the aqueous Fremy's salt solution.

2.3.2 *First derivative line shape vs. modulation power.*

For this procedure, a series of LFESR scans were made of a 15 mL aqueous sample of ~100 μM Fremy's salt, at different modulation power settings (See Results, Figure 2-3). The modulation power setting was incremented through a range of values, by adjusting the power output level of the AF power amplifier, A3, to values of: 0.01, 0.03, 0.1, 0.3, 1, 3, and 10 watts / 8 ohms. These values were chosen to correspond to the dB scale of the front panel power output meter of the AF amplifier. Adjustments were made by manually adjusting the output level knob on the front panel of A3, until the desired output level was indicated by the corresponding front panel power output meter.

The scans were each made in the region of the central line of the 3-line spectrum of Fremy's salt, by scanning from 8.400 to 9.000 mT, in steps of 0.005 mT. The peak-to-peak data for each scan was used to determine the relationships between

peak-to-peak amplitude vs. modulation power (See Results, Figure 2-4), and peak-to-peak width vs. modulation power (See Results, Figures 2-5a and 2-5b).

2.3.3 *First derivative line shape vs. RF power attenuation.*

For this procedure, a series of LFESR scans were made of a 15 mL aqueous sample of ~150 μM Fremy's salt in 10% sodium carbonate (See Results, Figure 2-6). Each scan of the series was made at a different level of RF power attenuation. The RF power attenuation was adjusted to values of: 10, 12, 14, ... 30, 35 and 40, using the RF attenuators, R3 and R4, at the front panel of the LFESR Control Panel (See Appendix B, Figure B1). The peak-to-peak data for each scan was used to determine the relationships between peak-to-peak amplitude Vs. RF power attenuation (See Results, Figures 2-7a and b), and peak-to-peak derivative width Vs. RF power attenuation (See Results, Figures 2-8).

2.3.4 *Line intensity vs. spin concentration.*

For this procedure, a series of 15 mL aqueous Fremy's salt spin standards were prepared to have concentrations of 1.20 mM, 600 μM, 300 μM, 150 μM, and 75 μM. Each standard was scanned four times, from 8.600 mT to 9.000 mT, in steps of 0.005 mT (See Results, Figure 2-9). The RF power attenuation, and modulation power setting were each set to the values that corresponded to the maximum sensitivity, as

determined by the previous procedures. The RF attenuation was set at 10 dB, and the modulation power was set at 1 watt / 8 ohms. The concentration stability of each standard was monitored, by measuring the optical absorbance of the standard at 248 nm, immediately before and after each set of four scans. The scan range was chosen to exclusively capture the center line of the characteristic three-line spectrum of the Fremy's salt solutions. The data from each scan was saved to a disk file, and later analyzed to determine the relationship between spin concentration and line intensity for the Fremy's salt spin standards (See Results, Figure 2-11).

2.3.5 *SNR vs. spin concentration.*

This procedure is identical to that described for "Signal Intensity Vs. Analyte Concentration" The SNR from each scan was calculated, and used to determine the relationship between SNR and spin concentration (See Results, Figure 2-12).

2.3.6 *SNR vs. signal averaging.*

For this procedure, a 300 μM aqueous Fremy's solution was prepared in 10% sodium carbonate, using HPLC water. Spectra were acquired in the region of the center peak, from 8.4 to 9.2 mT, in 0.01 mT steps, at a frequency of 249.436 MHz.

Three sets of spectra were collected. Each set consisted of two spectra. The first spectrum in each set served as the control, that showed what the SNR would be

without using any signal averaging (i.e. with n = 1). The second spectrum in each set served as the test, that showed what the SNR what be if signal averaging was used (i.e. with n > 1). For each set of spectra, a probe-tuning sweep was performed, immediately followed by the control and test spectra acquisition. The reason for breaking the procedure into 3 separate sets, with each preceded by a probe-tuning sweep, was to control the effect of probe-detuning, which may occur over sufficiently long periods of time. The detuning effect can lead to a change in the SNR of a spectrum with all other parameters held constant (See Results, Figures 2-13 through 2-15).

2.4 Results and Discussion

2.4.1 *LFESR scan of Fremy's salt.*

Figure 2-2 represents a first derivative LFESR spectrum of a 75 μM solution of Fremy's salt, prepared in 10% aqueous sodium carbonate. The spectrum consists of three first derivative lines located at 7.515, 8.735 and 10.155 mT. The separation between the first and second lines is 1.220 mT. The separation between the second and third lines is 1.420 mT. The relative intensities of the first, second and third lines of the spectrum were determined by double integration of the first derivative data to be 1.05, 1.00 and 1.03, respectively.

The spectrum shows good peak-to-peak symmetry (i.e. the positive and negative responses of each peak are symmetric). This indicates that the length of the Phase Adjusting Loop, P1, is properly adjusted. The lack of symmetry in the splitting pattern is do to the non linear spacing that results in the energy levels for the nitroxide radical at 250 MHz (See Appendix D). Finally, the relative intensities of the peaks are within 5% of each other. Peak intensities of 1:1:1 are expected for Fremy's salt, since nitrogen has an spin = 1 nucleus. The spectral results agree well with the literature, which reports three equally intense lines, separated by 1.31 mT (Wertz and Bolton, 1972).

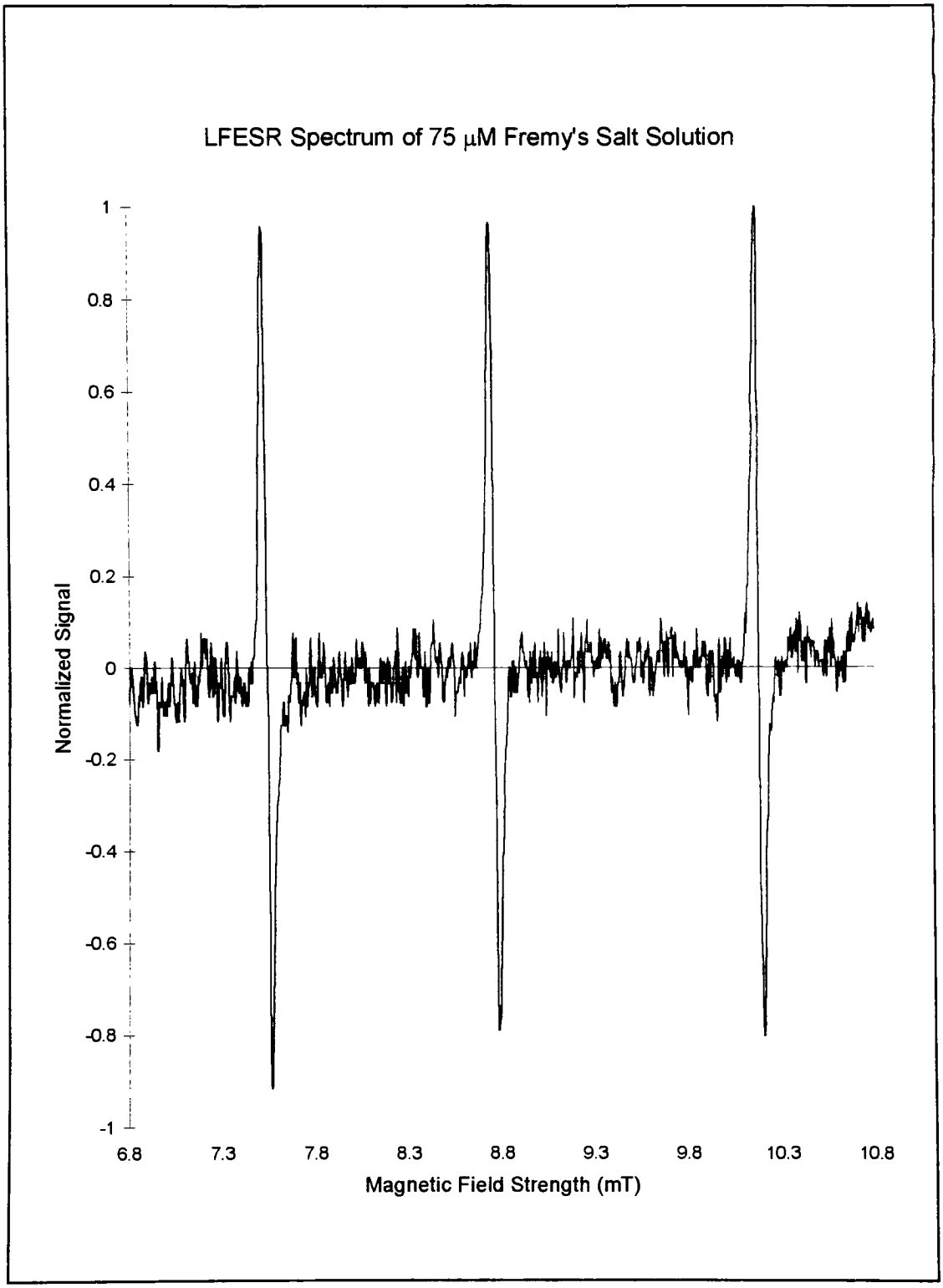


Figure 2-2 The LFESR spectrum of a 75 μM aqueous solution of Fremy's salt, in 10% sodium carbonate.

2.4.2 *First derivative line shape vs. modulation power.*

Figure 2-3 represents an overlay of seven separate spectral scans of 150 μM Fremy's salt, prepared in 10% aqueous sodium carbonate. Each scan corresponds to a progressively higher modulation power setting, as indicated. This figure illustrates the overall effect of modulation power on spectral line shape. The figure also defines peak-to-peak derivative amplitude, A_{pp}, and peak-to-peak derivative width, B_{pp}.

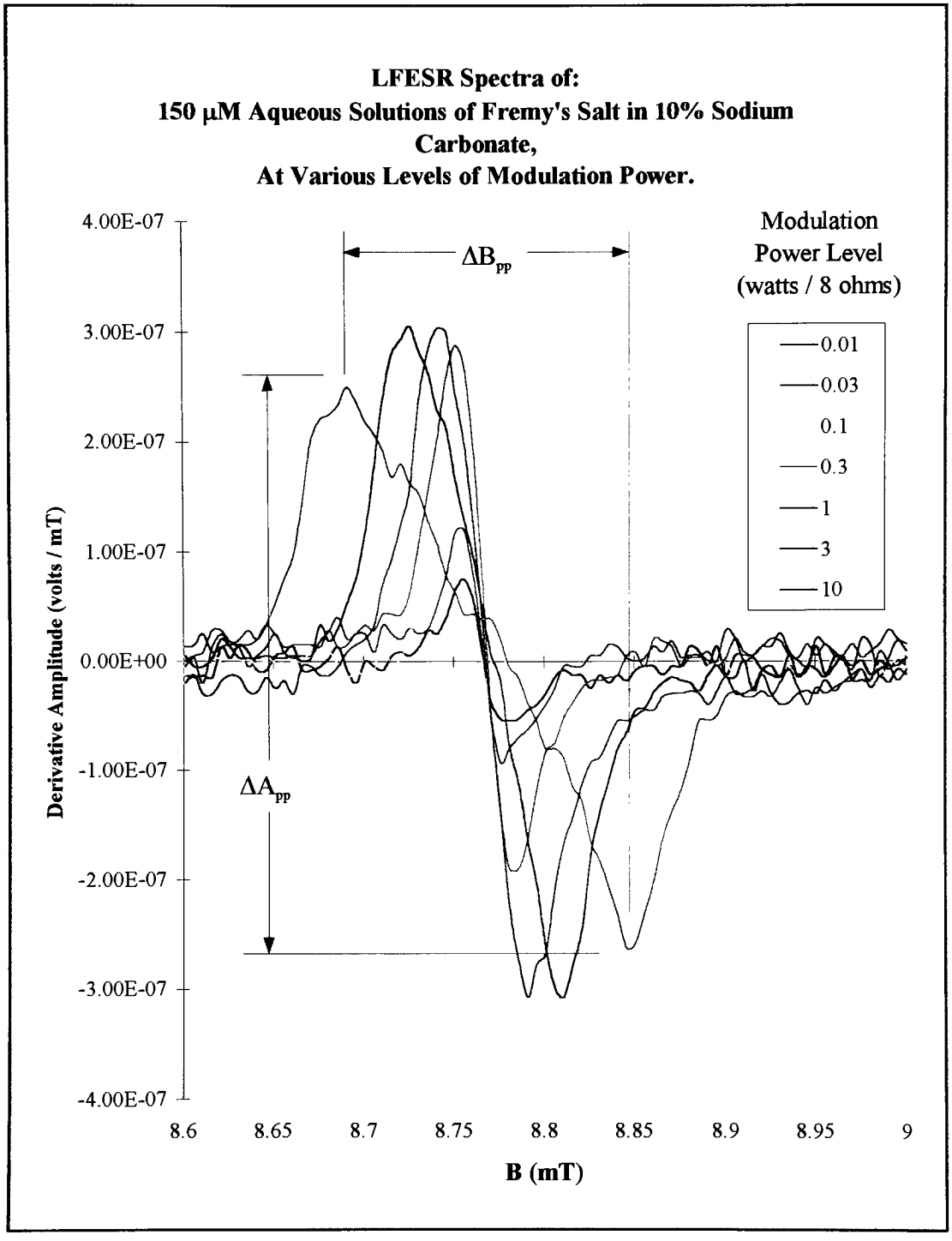


Figure 2-3 First derivative line shape vs. modulation power. Note the effects of modulation power on peak-to-peak derivative amplitude, A_{pp}, and peak-to-peak derivative width, B_{pp}.

Figure 2-4 represents a plot of A_{pp} vs. modulation power. As shown by the plot, A_{pp} increases with respect to increases in modulation power, for modulation power settings ranging from 0.01 to 1.00 watts / 8 ohms. The value of A_{pp} reaches an apparent maximum at a modulation power setting of approximately 1 watt / 8 ohms. At modulation power settings exceeding 1 watt / 8 ohms, A_{pp} decreases.

Figure 2-5a is a plot of B_{pp} vs. modulation power. Figure 2-5b is an expanded view of the B_{pp} vs. modulation power plot for the range of 0.01 to 10 watts / 8 ohms. The relative peak-to-peak derivative amplitude remains approximately constant for the range of 0.01 to 0.1 watts / 8 ohms. As the modulation power setting is increased through the range of 0.1 to 100 watts / 8 ohms, the peak-to-peak line width experiences significant broadening. The constant peak-to-peak width of 0.25 mT represents the true peak-to-peak line width (Wertz and Bolton, 1972).

From these results, the optimum setting for modulation power will depend on whether maximum sensitivity or maximum resolution is desired. A value of 1 watt / 8 ohms results in maximum sensitivity, but results in a sacrifice in resolution, with relative line broadening of 200 percent. A value of 0.1 results in maximum sensitivity obtained without the loss of resolution, with relative line broadening of less than 2 percent, but results in a sacrifice of sensitivity, with a 44 percent decrease in peak-to-peak derivative amplitude.

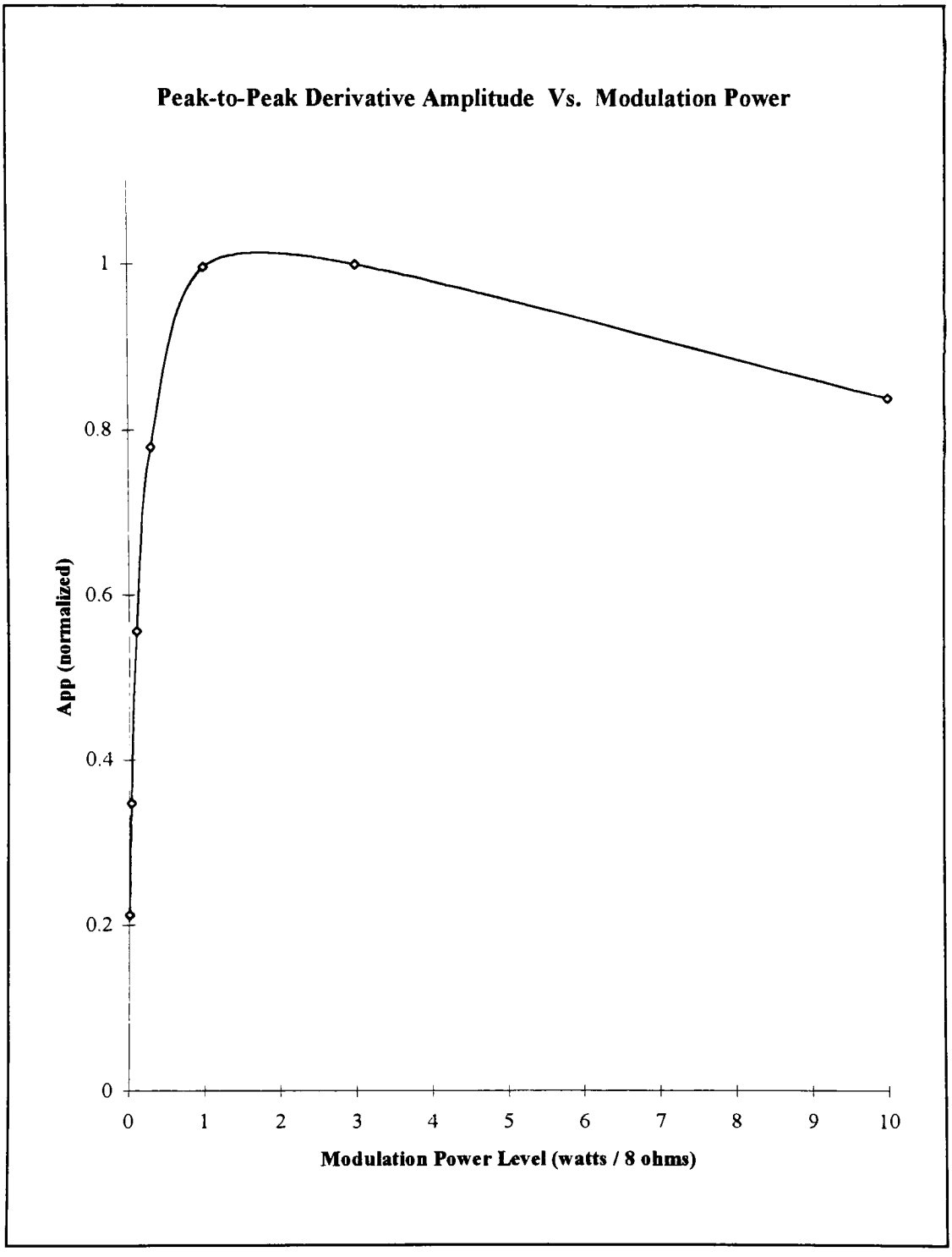


Figure 2-4 A plot of normalized peak-to-peak derivative amplitude, A_{pp}, vs. modulation power level. A maximum A_{pp} occurs at a modulation power level of ~1.5 watts / 8 ohms.

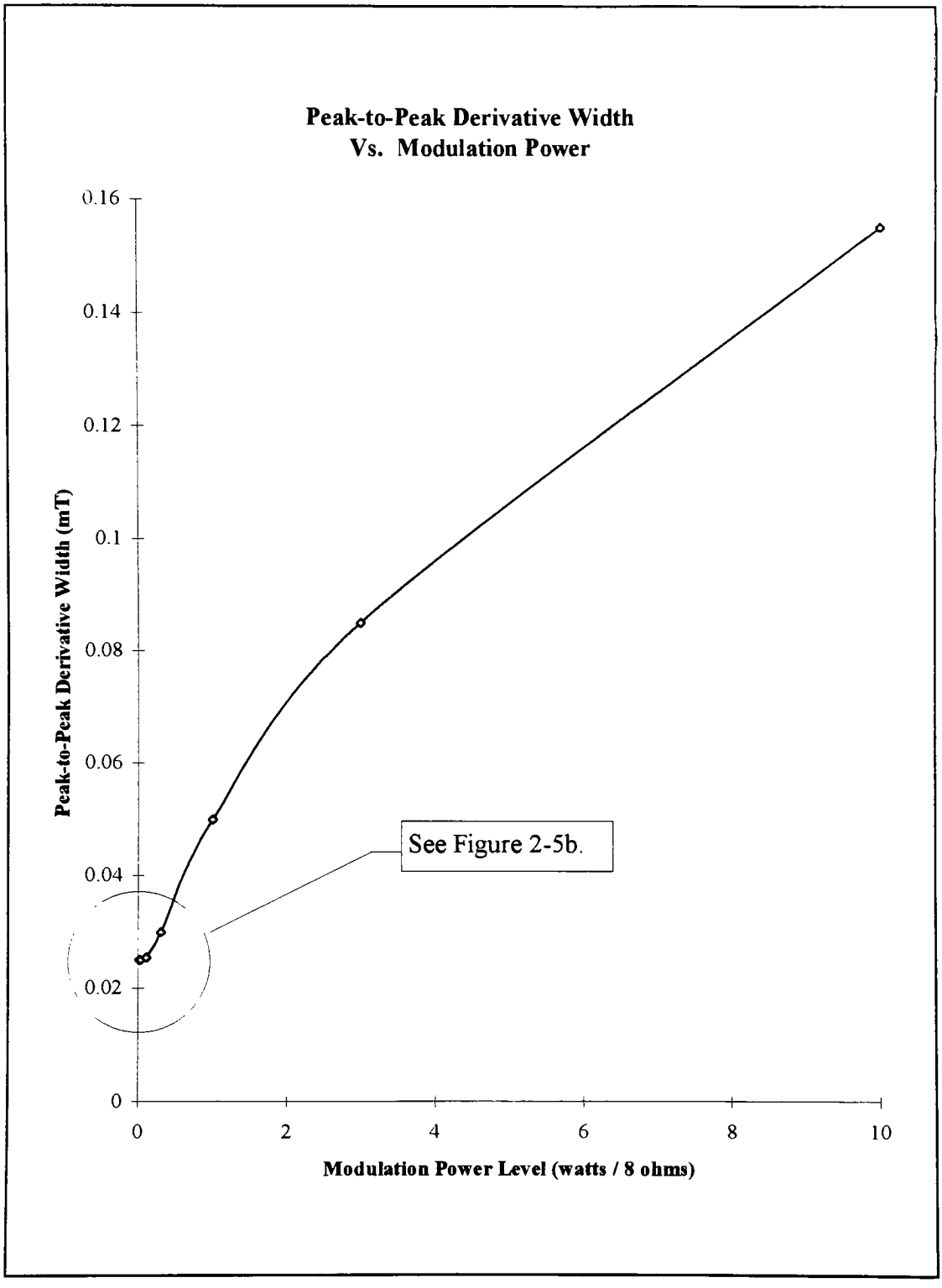


Figure 2-5a A plot of relative peak-to-peak derivative width, B_{pp}, vs. modulation power level.

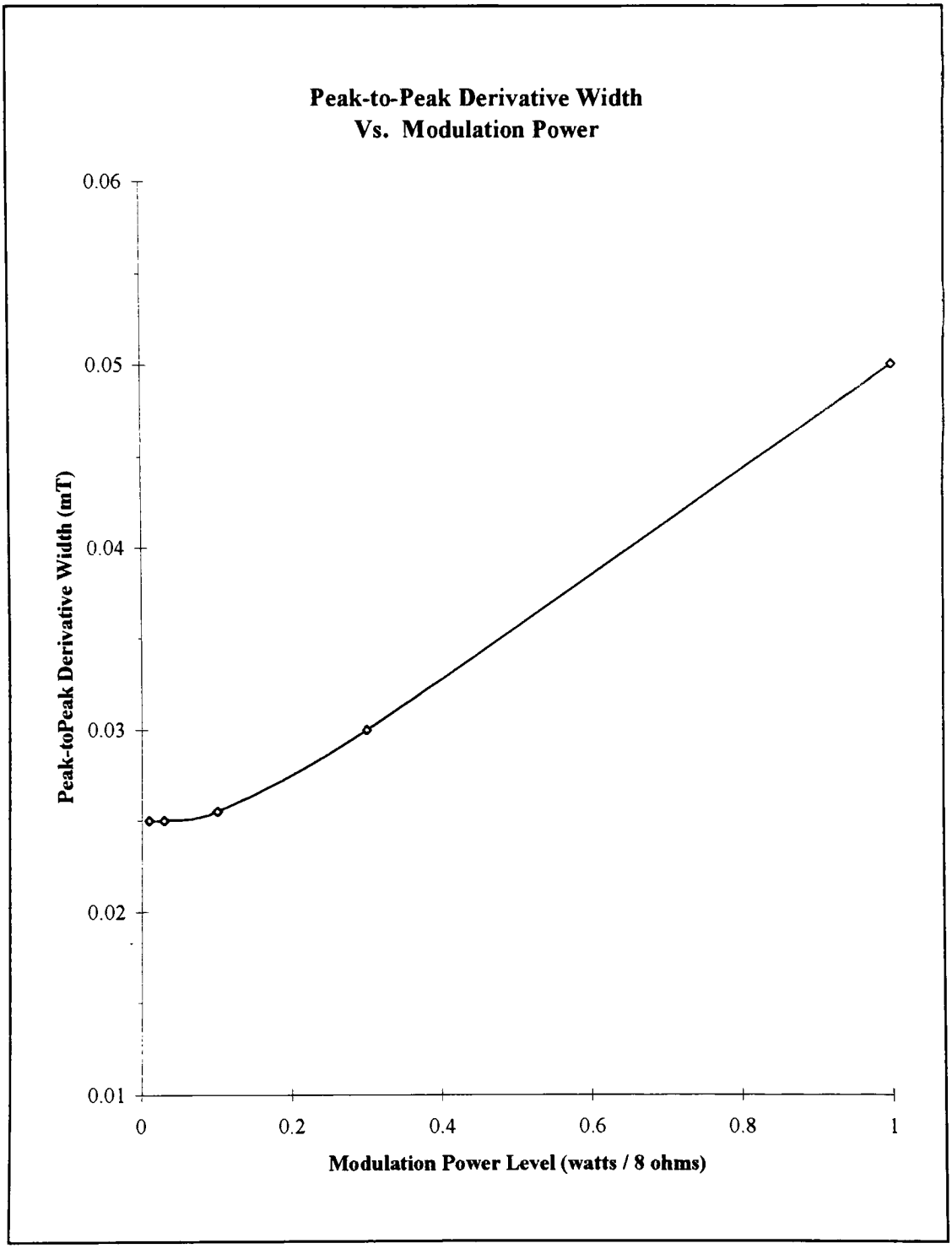


Figure 2-5b An expanded view of the 0.01 to 1 watt / 8 ohm region of the plot of relative peak-to-peak derivative width, B_{pp}, vs. modulation power level, illustrated in Figure 2-5a.

2.4.3 *First derivative line shape vs. RF power attenuation.*

Figure 2-6 represents an overlay of thirteen separate spectral scans of 150 mM Fremy's salt, prepared in 10% aqueous sodium carbonate. Each scan corresponds to a progressively higher setting of RF power attenuation, as indicated. This figure illustrates the overall effect of RF power attenuation on spectral line shape.

Figure 2-7a is a plot of App vs. RF power attenuation. When there is no saturation, App2 is proportional to P (Wertz and Bolton, 1972). Note that a dB of power attenuation is equal to 10 Log(P/Po), where Po is a reference power. Therefore, P/Po is equal to 10attenuation/10. Figure 2-7b is a plot of App2 vs. P/Po. Although the result shows a slight degree of saturation occurring at higher values of P/Po, as evidenced by the deviation from linearity of the data, the saturation effect is minimal, and not expected to have a negative effect on performance. Note that the RF power attenuation setting of 10 dB corresponds to the minimum limit of attenuation, as determined by the safe operating limits of the RF bridge components.

Figure 2-8 is a plot of peak-to-peak width vs. RF power attenuation. As RF power attenuation is varied from 10 to 40 dB, the peak-to-peak width remains constant as shown. This results shows that the line width is undistorted by the RF power for the tested range.

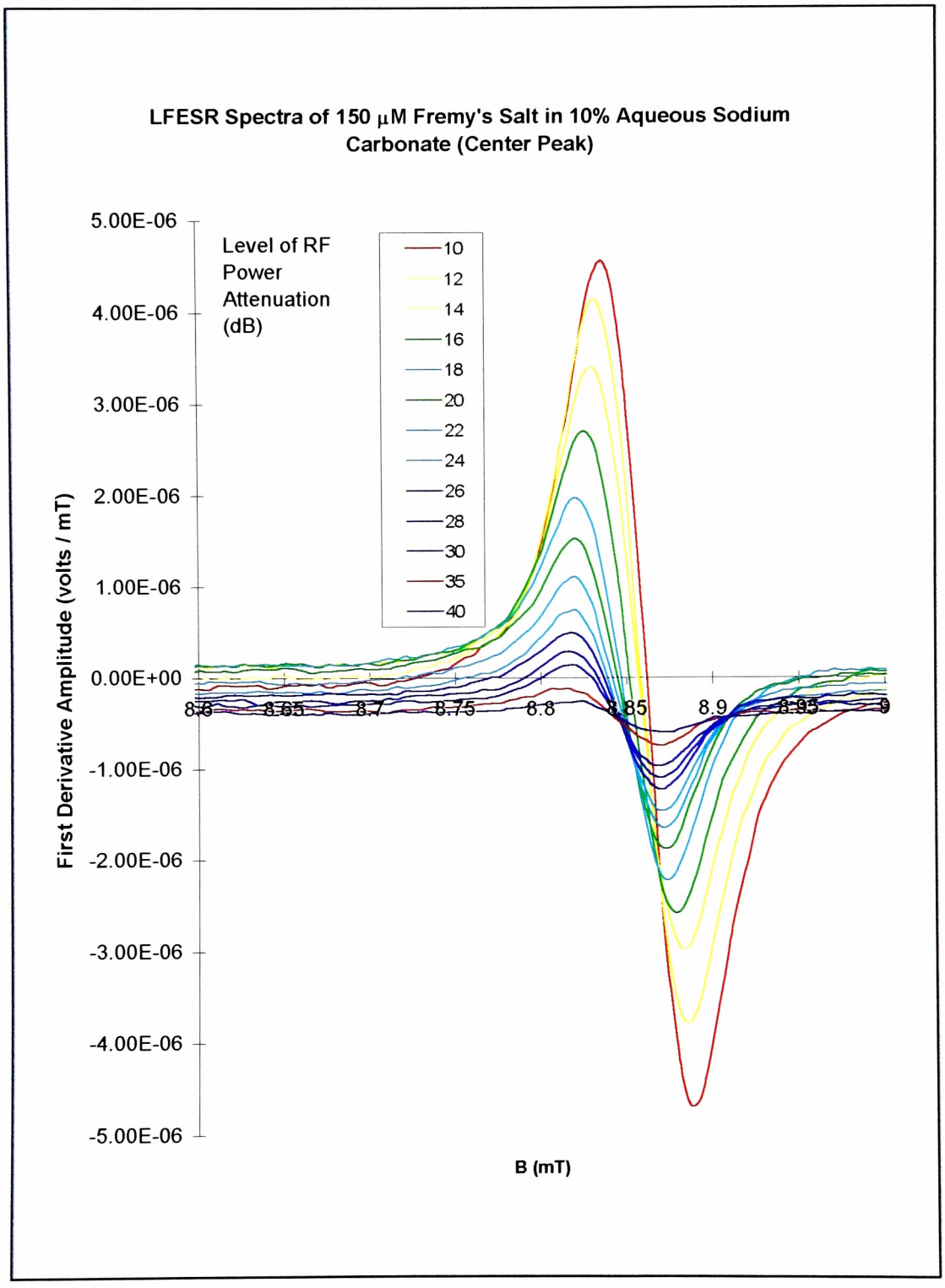


Figure 2-6 LFESR Spectra of a 150 μM aqueous solution of Fremy's salt, scanned at various levels of RF power attenuation, as indicated.

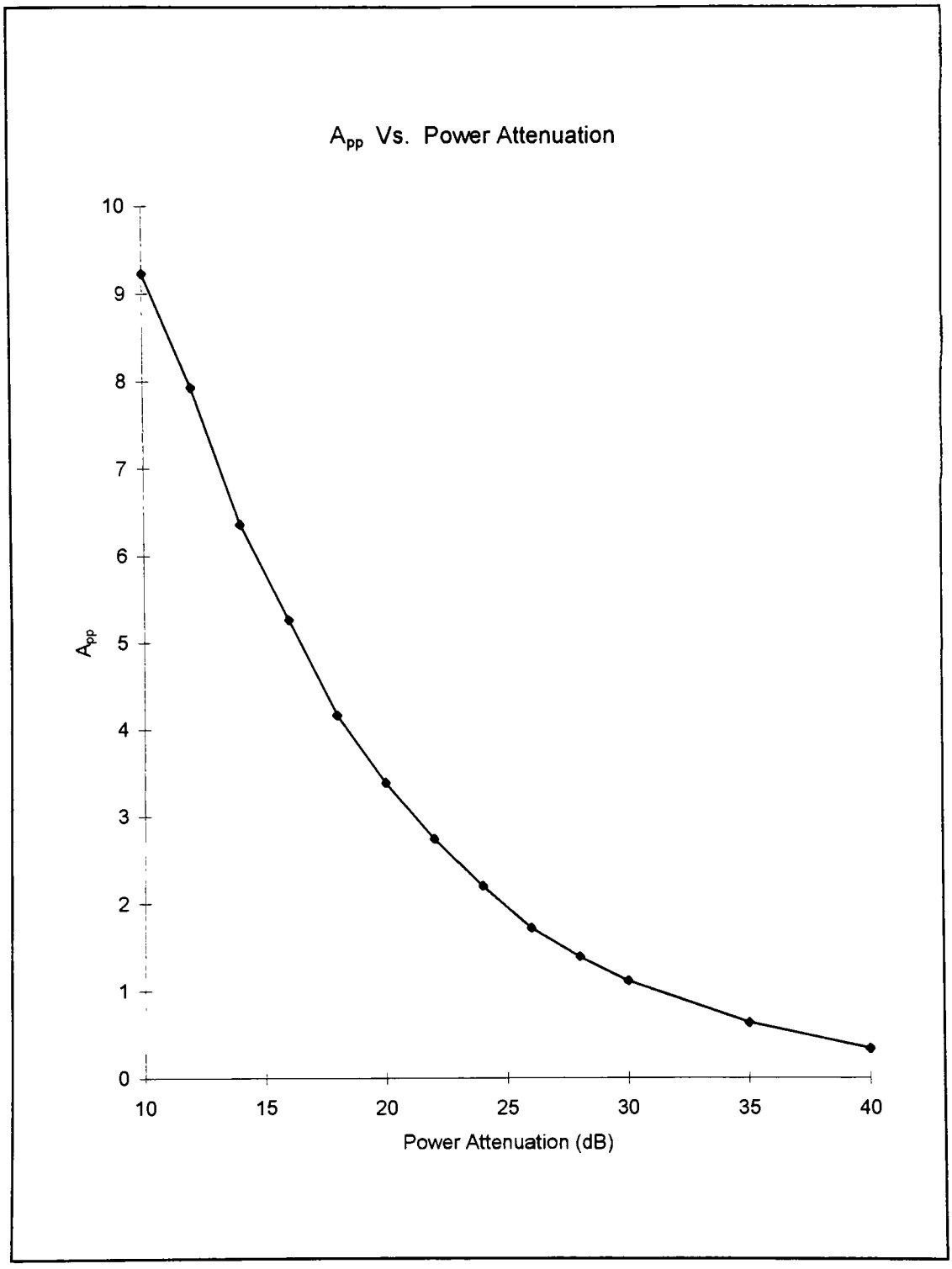


Figure 2-7a A plot of A_{pp} vs. RF power attenuation, for a 150 μM standard solution of Fremy's salt.

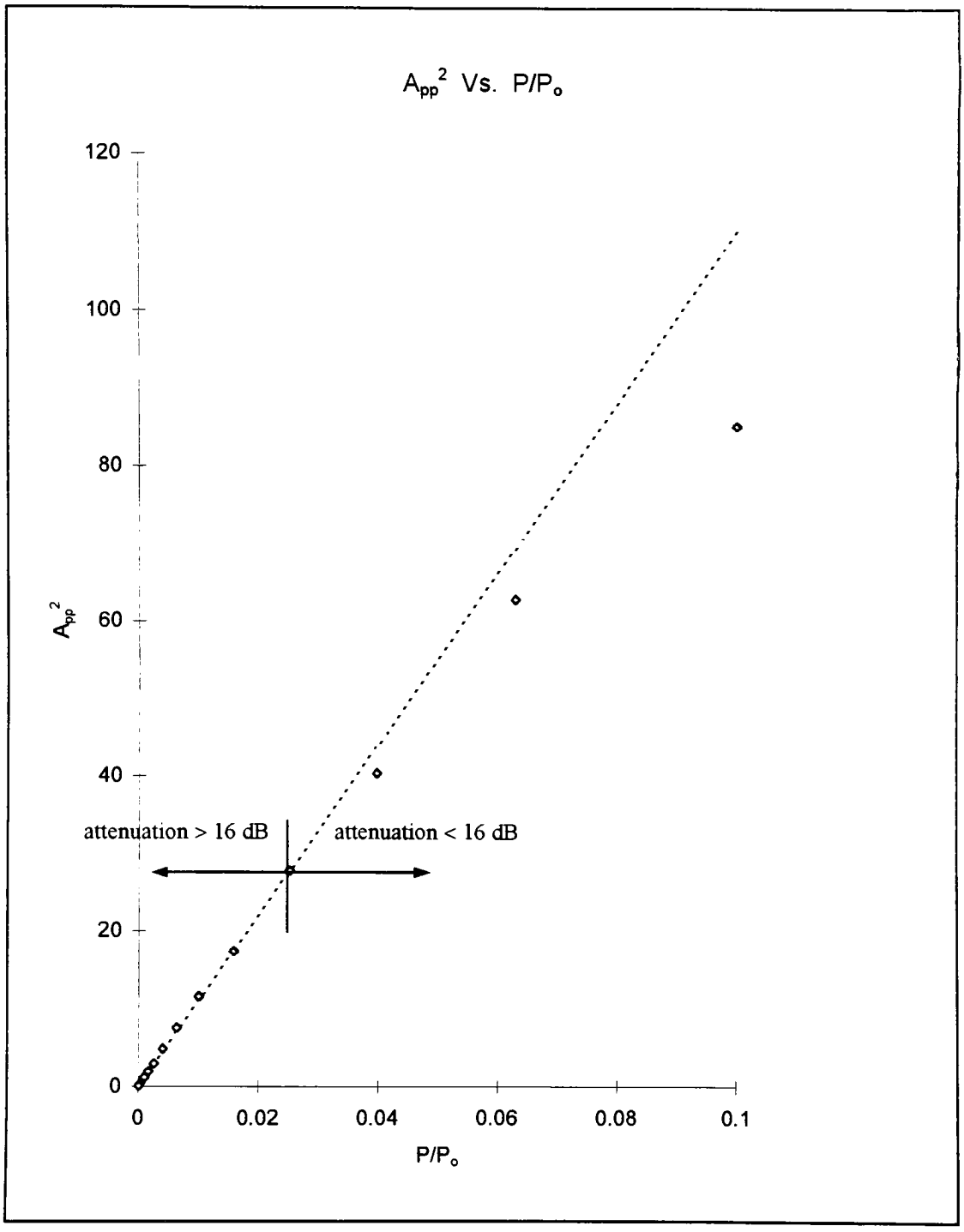


Figure 2-7b A plot of A_{pp}^2 vs. P/P_o, for a 150 μM standard solution of Fremy's salt. The data has a linear correlation coefficient ≥ 0.999, for attenuation settings ≥ 16 dB. The linear correlation coefficient decreases from 0.999 to 0.994 as the RF power attenuation is decreased from 16 to 10 dB.

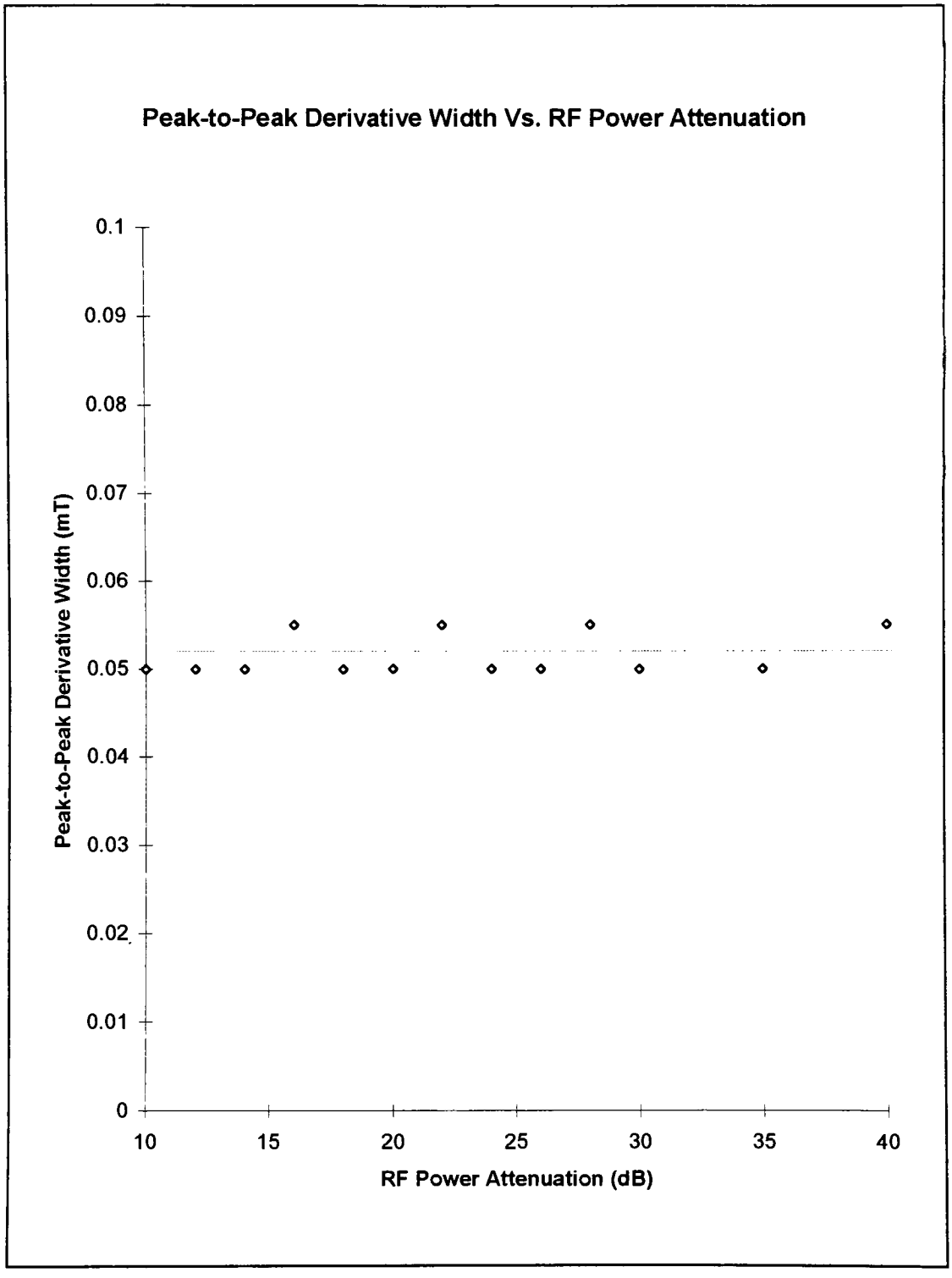


Figure 2-8 A plot of peak-to-peak derivative width vs. RF power attenuation, for a 150 μM standard solution of Fremy's salt.

2.4.4 *Line intensity vs. spin concentration.*

Figure 2-9 represents an overlay of first derivative spectral scans acquired from six separate 15 mL Fremy's salt spin concentration standards. The six standards were prepared in aqueous sodium carbonate, to have concentrations of 2400, 2100, 300, 150 and 75 μM.

Figure 2-10 represents an overlay of the absorption lines that correspond to the first derivative lines in Figure 2-9. The absorption lines are obtained by numerically integrating the first derivative data in Figure 2-9, over the scan range of 8.600 to 9.000 mT. The intensity of a spectral line is being treated her as the area under the absorption line. In order to relate the intensity of spectral lines to the absolute number of spins in each corresponding standard, the area under each absorption line is determined by integrating the absorption results in Figure 2-10 over the scan range of 8.600 to 9.000 mT.

Figure 2-11 is a plot of signal intensity vs. absolute number of spins. The plot shows a linear relationship between the signal intensity and absolute number of spins for the range of standards being used. This result is expected to be linear for the low concentrations being used here. However, at sufficiently high concentrations, departures from linearity are expected, due to inhomogeneous line broadening resulting from spin exchange interactions.

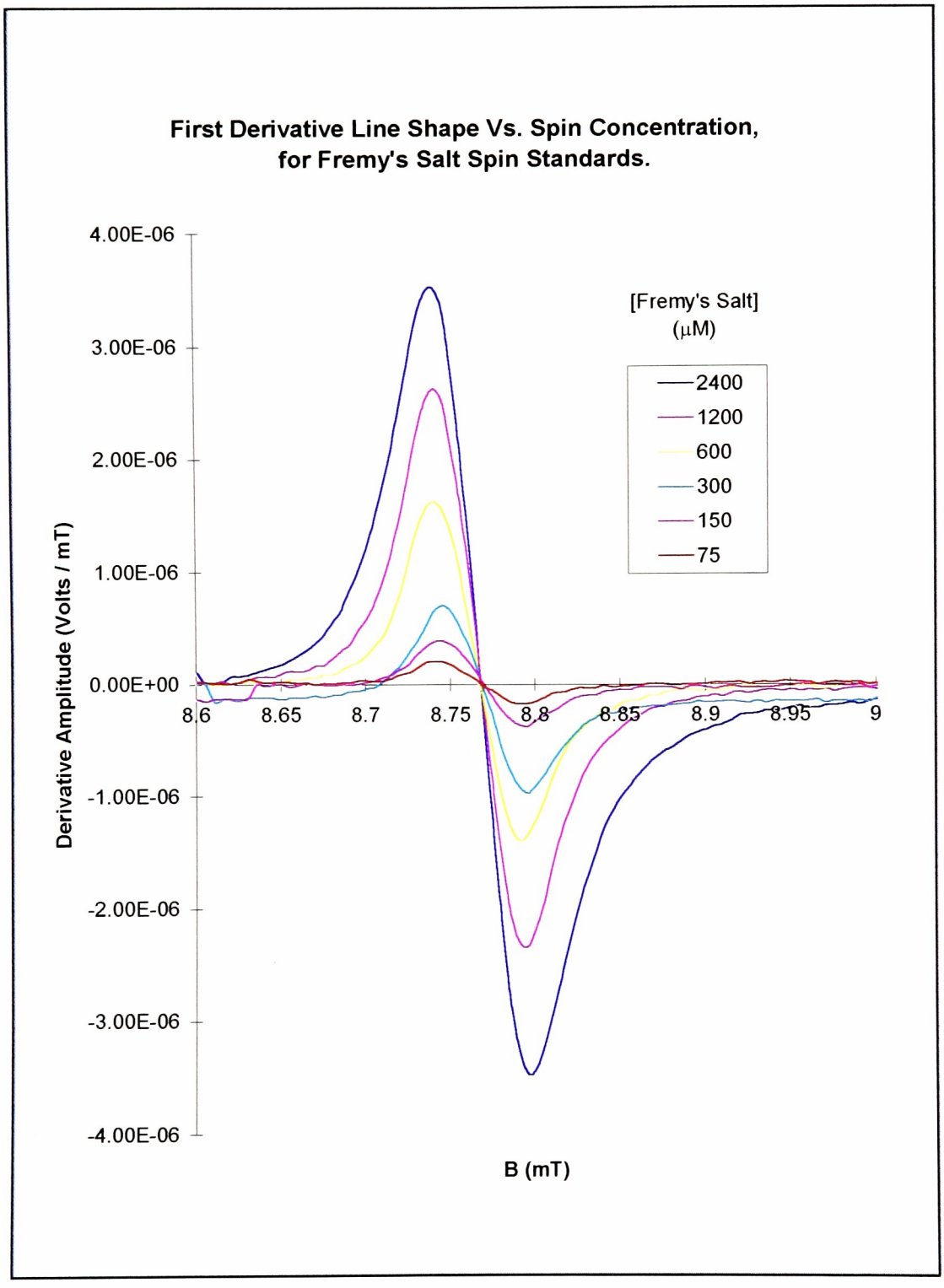


Figure 2-9 First derivative LFESR spectra of 15 mL Fremy's salt spin standards, prepared to have concentrations of 75, 150, 300, 600 and 1200 μM.

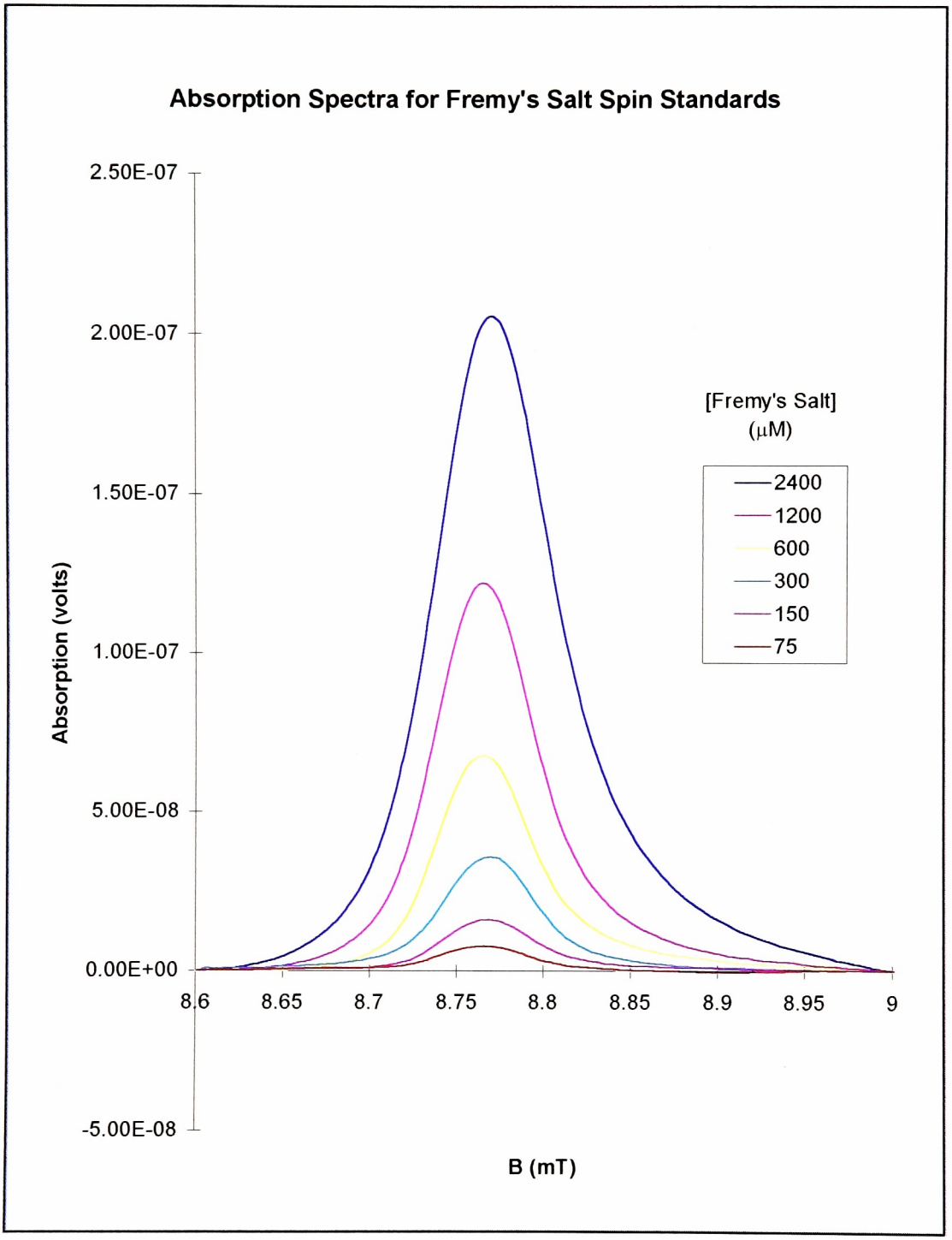


Figure 2-10 LFESR absorption spectra of 15 mL Fremy's salt spin standards, prepared to have concentrations of 75, 150, 300, 600 and 1200 μM. The absorption lines were obtained by integrating the experimentally obtained first derivative spectra of Figure 2-9.

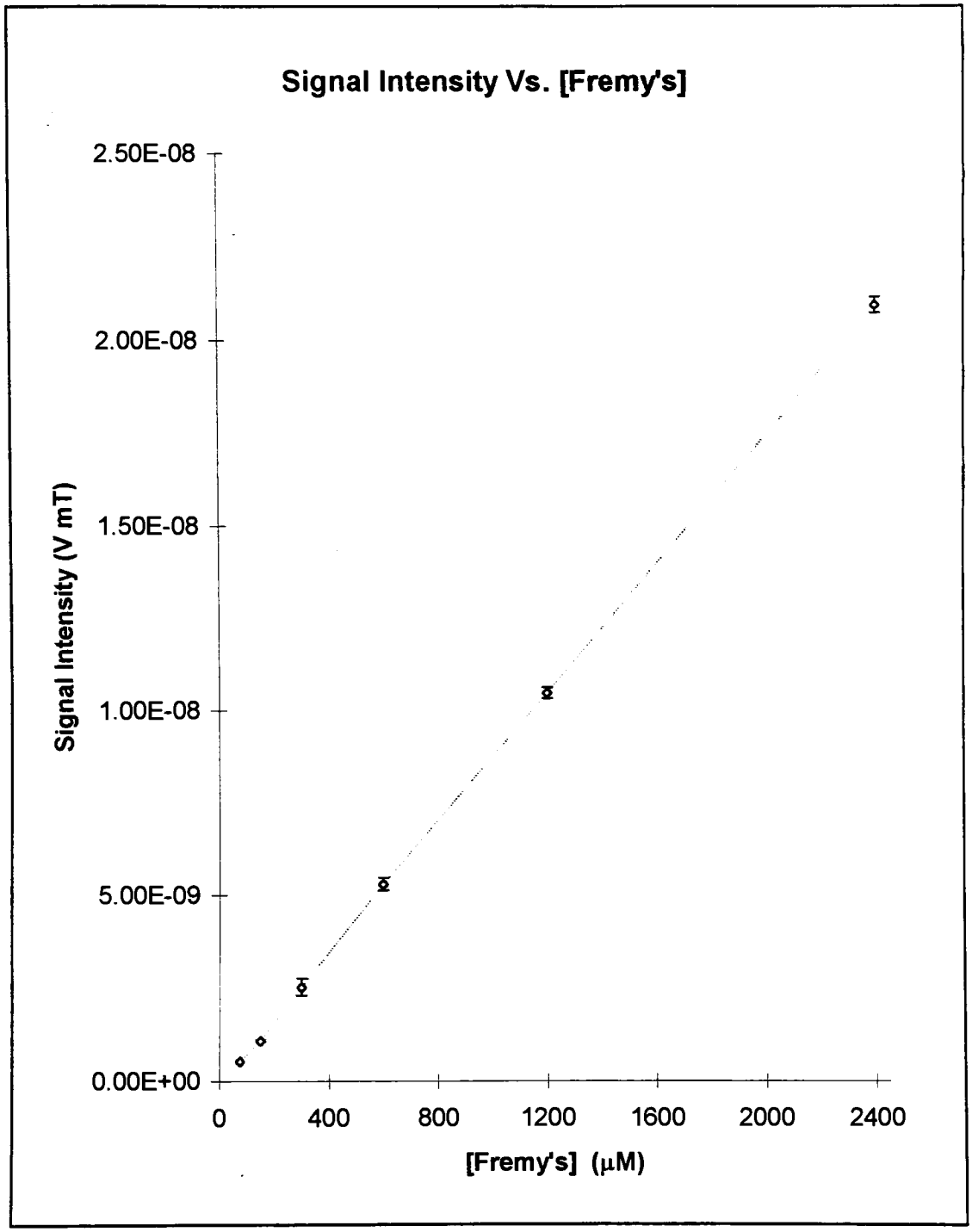


Figure 2-11 A plot of signal intensity vs. spin concentration, for Fremy's salt spin standards, prepared to have concentrations of 75, 150, 300, 600, 1200 and 2400 μM. Signal intensities were obtained by integrating the absorption lines illustrated in Figure 2-9.

2.4.5 *SNR vs. spin concentration - line width dependent sensitivity limit.*

Figure 2-12 is a plot of SNR vs. spin concentration, and is based on the analysis of the same spectral data illustrated in Figure 2-9. This result shows that in order to have an SNR of 10 or greater, the concentration of Fremy's salt must be at least 75 μM. This concentration is equivalent to 6.77×10^{17} spins in the 15 mL sample. Since the spectrum of Fremy's salt consists of three equally intense lines, the number of spins that correspond to each line in the spectrum is equal to 6.77×10^{17} spins / 3, or 2.26×10^{17} spins. The lines have a width of 0.05 mT under the conditions of modulation amplitude being used in this study (1 watt / 8 ohms). Therefore, the **line width dependent detection limit** of the spectrometer is equal to:

2.26×10^{17} spins / 0.05 mT = **4.52×10^{21} spins / T**.

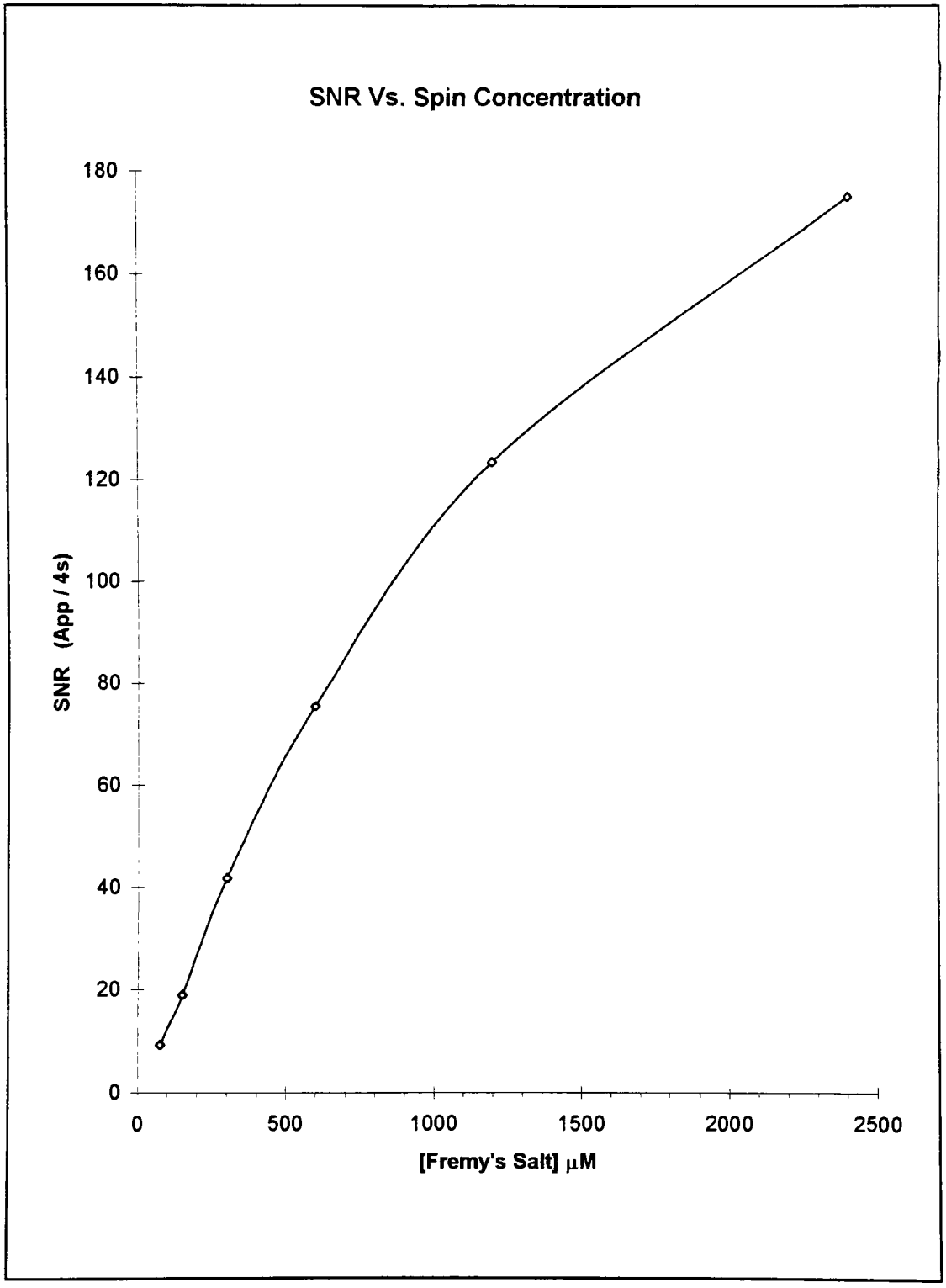


Figure 2-12 A plot of SNR vs. spin concentration, for Fremy's salt spin standards prepared to have concentrations of 75, 150, 300, 600, 1200 and 2400 μM.

2.4.6 *SNR vs. signal averaging.*

Figures 2-13 through 2-15 illustrate the effect of signal averaging on SNR. Each figure consists of a pair of spectra, where the first spectrum in each pair is scanned with no signal averaging (n = 1) and used as a reference, and the second spectrum in each of the three pairs is scanned using progressively higher levels of signal averaging (n = 4, 9 and 16). The signal to noise ratios of each scan are determined as the ratio of peak-to-peak amplitude, App, and the standard deviation in the base-line data of the scan. The standard deviation is measured for the data in the 8.4 to 8.6 mT range of each scan. For each pair of scans, a factor of improvement in the SNR is determined with respect to the value of n. The factor of improvement is calculated as the ratio of the SNR with signal averaging to the $SNR_{n>1}$ without signal averaging, $SNR_{n=1}$

Figure 2-16 is a plot of the factor of improvement in SNR vs. the value of n. Figure 15 also shows the predicted factor of improvement, calculated as $n^{1/2}$ The test results seem to agree well with the predicted results, better correlation at for high values of n (n = 16) than at low values of n (n = 4). From this result, a significant improvement in SNR, and therefore in sensitivity, can be gained by employing the use of signal averaging. However, signal averaging requires a longer period of time for spectral data acquisition, which may not be permitted in some cases involving short-lived species.

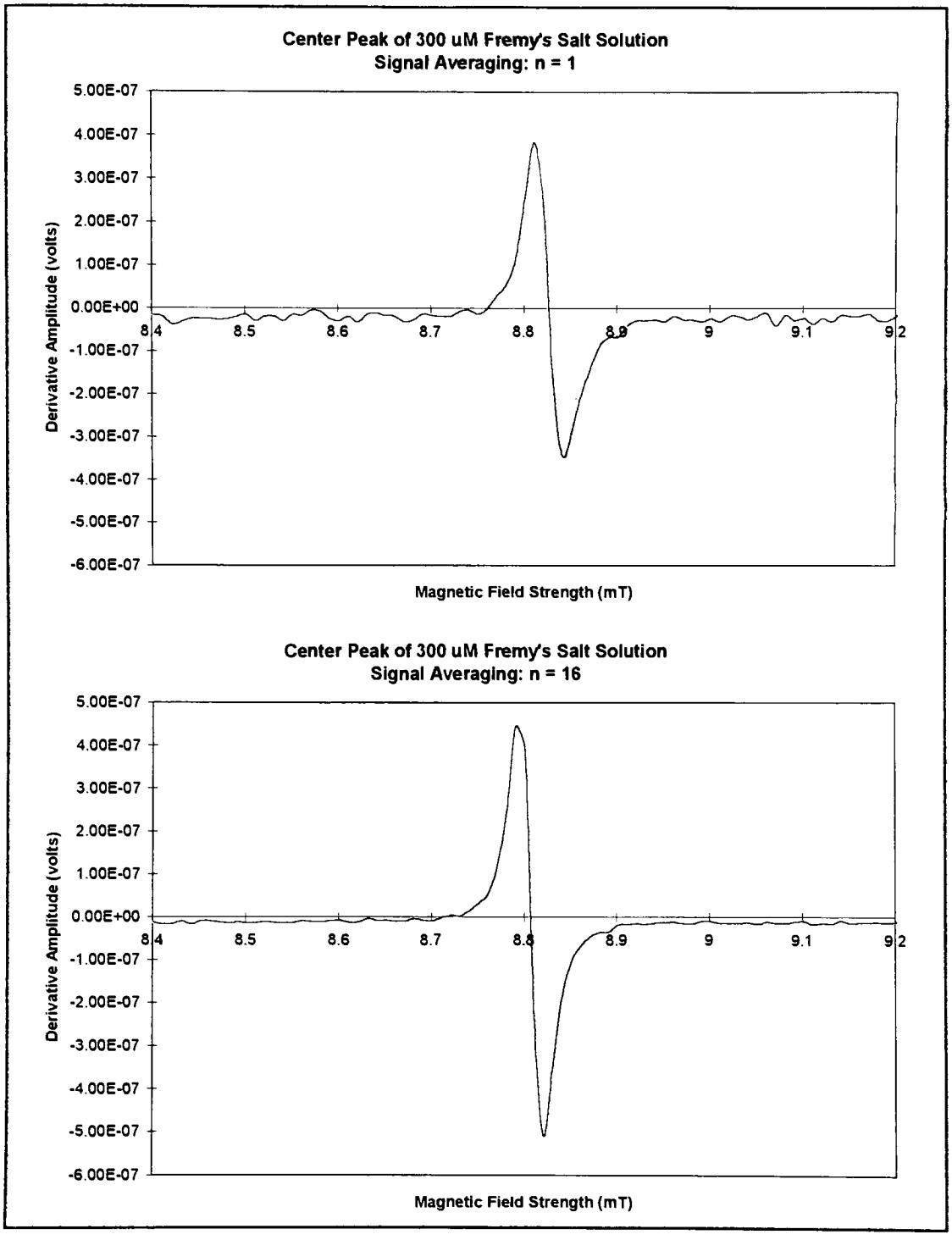


Figure 2-13 The Effect of Signal Averaging on SNR, where n is increased from 1 to 4. (a) Spectra with no signal averaging, where n = 1, and SNR = 56.5; (b) Spectra with signal averaging, where n = 4, and SNR = 147. The SNR improves by a factor of 2.60.

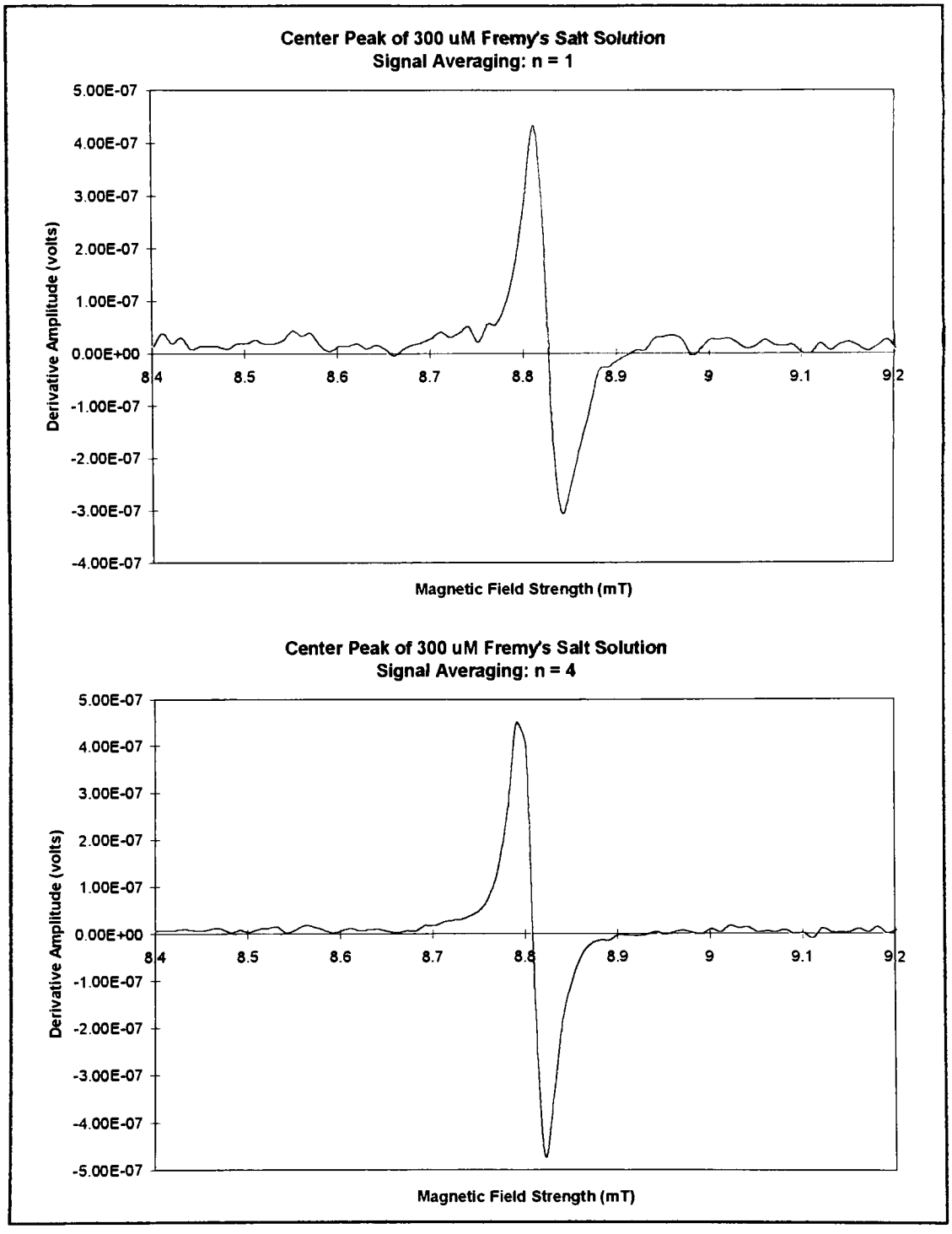


Figure 2-14 The Effect of Signal Averaging on SNR, where n is increased from 1 to 9. (a) Spectra with no signal averaging, where n = 1, and SNR = 66.8; (b) Spectra with signal averaging, where n = 9, and SNR = 214. The SNR is improved by a factor of 3.20.

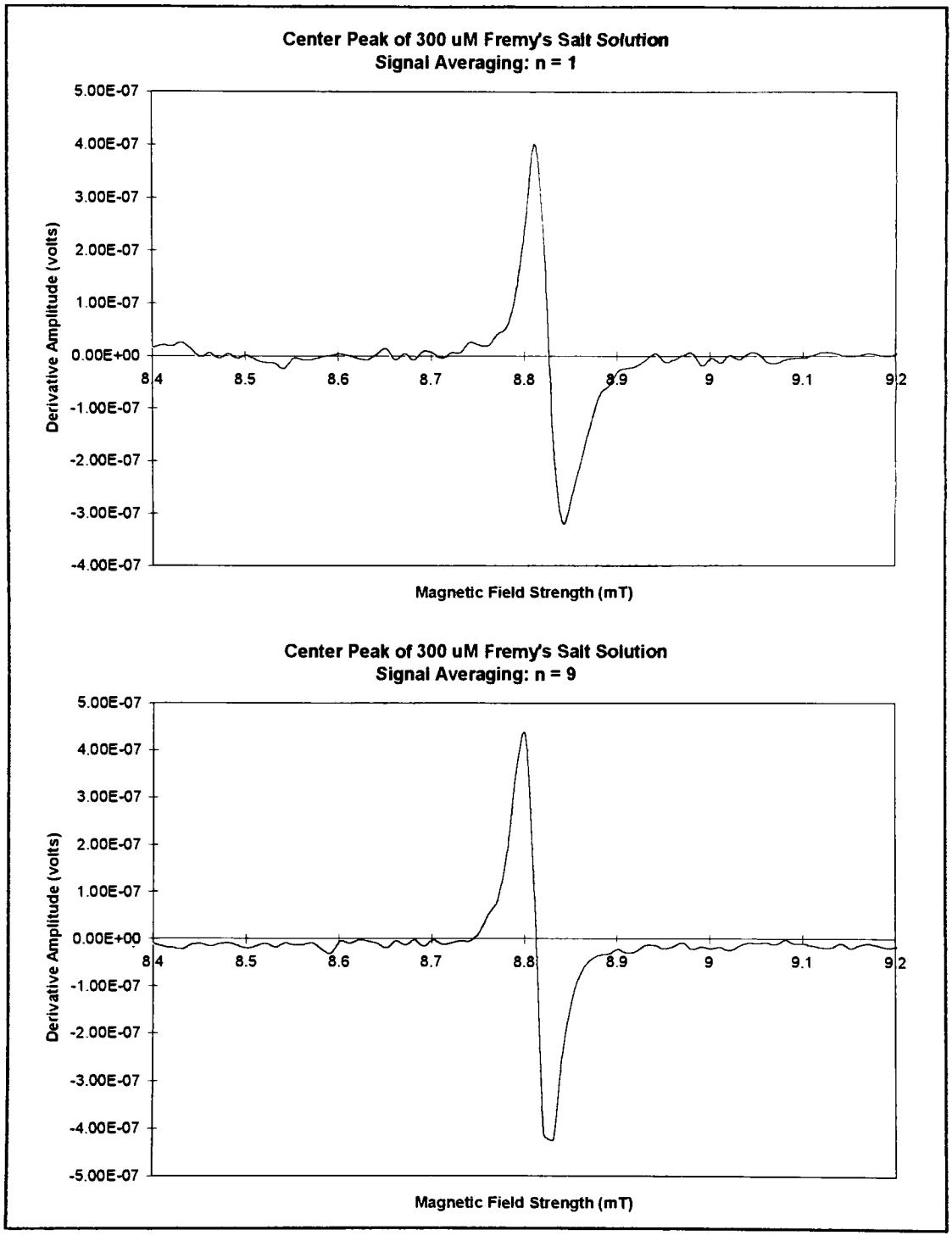


Figure 2-15 The Effect of Signal Averaging on SNR, where n is increased from 1 to 16. (a) Spectra with no signal averaging, where n = 1, and SNR = 91.0; (b) Spectra with signal averaging, where n = 16, and SNR = 370. The SNR improves by a factor of 4.07.

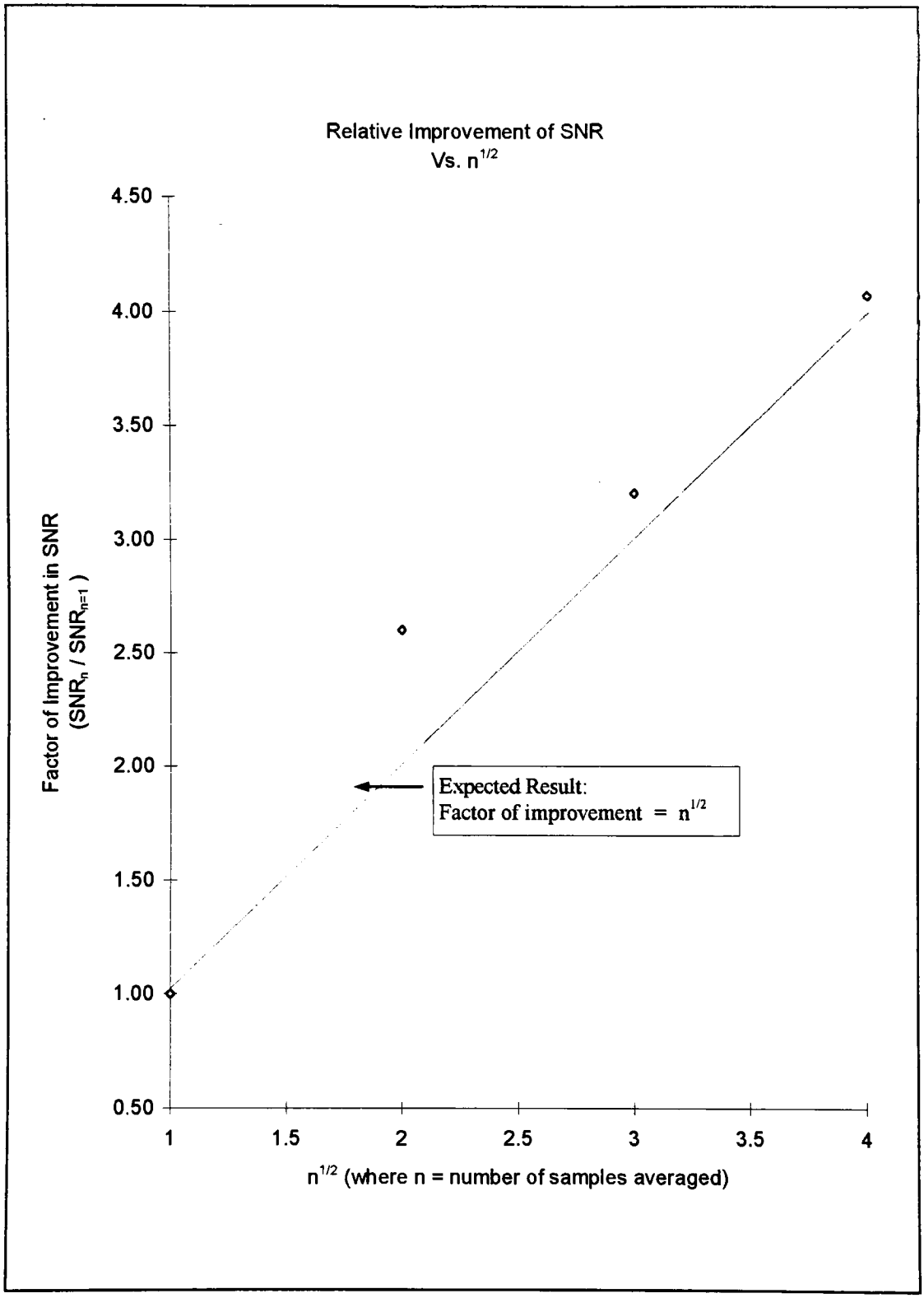


Figure 2-16 A plot of relative SNR vs. $n^{1/2}$, where n is the number of samples averaged for each data point of a spectrum of a 75 μM Fremy's salt spin standard.

2.4.7 *Summary of results.*

Table 2-1 LFESR performance characterization test results.

Test	Range	Result	Comments
A_{pp} Vs. Field Modulation Power Setting	0.01 to 10 watt / 8 ohm	A maximum A_{pp} occurs at 1 watt / 8 ohm.	For maximum sensitivity, operate at 1 watt / 8 ohm. (Note: This setting results in a 2x distortion of line width.)
B_{pp} Vs. Field Modulation Power Setting	0.01 to 10 watt / 8 ohm	Line width is constant at levels < 0.1 watt / 8 ohm. Distortion of line width occurs at levels > 0.1 watt / 8 ohm.	For maximum sensitivity with no distortion of line width, operate at 0.1 watt / 8 ohm.
A_{pp} Vs. RF Power Attenuation Setting	10 to 40 dB	A maximum A_{pp} occurs at a setting of 10 dB.	Maximum sensitivity results at a setting of 10 dB. 10 dB is the lower limit of RF power attenuation that can be safely used on the LFESR spectrometer in the current configuration, without resulting in component damage.
A_{pp}^2 Vs. P/P_o	10 to 40 dB	A_{pp}^2 is linearly proportional to P/P_o.	RF saturation does not occur within the tested range of 10 dB to 40 dB of RF power attenuation.
Signal Intensity Vs. [Fremy's]	75 to 2400 μM	Signal Intensity is linearly proportional to [Fremy's].	
SNR Vs. [Fremy's]	75 to 2400 μM	SNR ≥ 10 for [Fremy's] ≥ 75 μM.	Conditions: Field modulation power setting = 1 watt / 8 ohm. RF power attenuation = 10 dB. The LFESR spectrometer has a line width dependent detection limit of 4.52×10^{21} spins / T.
SNR Vs. $n^{1/2}$	n = 1, 4, 9 and 16	SNR is linearly proportional to $n^{1/2}$.	

Part III

LFESR detection of radicals generated by ultrasound - a preliminary study.

1. Introduction

Clinical ultrasound devices are used in a wide variety of applications, including diagnostic imaging and therapy (Fleischer, 1989, Swanson, 1990). While clinical ultrasound procedures are generally regarded as being a low health risk to patients, there is an interest in the chemical effects of ultrasound. Recent electron spin

resonance (ESR) studies have shown evidence that clinical ultrasound devices can be used to generate the biologically dangerous ·H and ·OH free radicals in aqueous solutions (Neta, 1980, Finkelstein, 1995).

The generation of radicals by ultrasound occurs through a process known as transient acoustic cavitation (Robertson, 1969, Knapp, 1970, Miller, 1987). Transient acoustic cavitation involves the formation, growth and violent implosion of vapor and/or gas filled cavities in liquids subjected to the dynamic pressure reductions of a sound field. The violent collapse of cavities produces intense local heating and pressures, with temperatures as high as 5000°C, pressures as high as 500 atmospheres, and heating and cooling rates exceeding 10^9 K/s (Suslick, 1990). While cavitaion may only result in the barely noticable heating of the bulk solution, the energy liberated from the cavities is often sufficient to result in the sonochemical cleavage of water molecules to form ·H and ·OH radicals.

One of the problems encountered in the ESR study of ·H and ·OH radicals is that the radicals are so highly reactive. It is difficult if not impossible to generate sufficiently high concentrations of the radicals, that exist for the period of time that is required for direct ESR detection. This makes it neccessary to employ the technique of spin-trapping. Spin trapping involves reacting the short lived radicals with a compound called a spin trap, to form a longer-lived spin trap adduct. While the spin trap is not a radical itself, the spin trap adduct is a radical, and is more readily observed

by ESR spectroscopy than the untrapped radical. The ESR spectra of a spin trap adduct can be used to identify the trapped radical (Berliner, 1979).

Recent studies (Makino, 1982, Finkelstein, 1995) involving ESR detection of ultrasound induced ·H and ·OH radicals, have utilized nitrone spin traps. Nitrones are not radicals, but react with unstable radicals to form relatively stable nitroxide radicals. The most commonly used nitrone spin trap is 5,5-dimethyl-1-pyroline-n-oxide, DMPO. DMPO reacts with ·H and ·OH radicals to form DMPO-H and DMPO-OH spin trap adducts. Each of these adducts have well known ESR spectra (Makino, 1982).

The aforementioned studies have utilized conventional ESR spectrometers, operating at about 9.5 GHz. A literature search has revealed no attempts to perform similar low frequency ESR studies. Thus sample size has been restricted, due to the strong dielectric absorption of 9.5 GHz radiation by water. The application of low frequency electron spin resonance (LFESR) spectroscopy to this general area of study may help to facilitate future *in situ* and *in vivo* studies of radicals produced by ultrasound (Hornak, 1991).

The goal of this project is to test the application of a low freqency ESR spectrometer to the detection of DMPO-H and DMPO-OH adducts produced by a 20 kHz ultrasound source. The defining difference in this study from others, is that a

large (15 mL) sample volume will be accomidated, since the spectrometer operates at ~248 MHz, instead of the typical 9.5 GHz operating frequency. The key problem to overcome will be that of the relatively low detection limit of the LFESR spectrometer. The LFESR spectrometer (Interfaced and characterized in Parts I and II of this document, respectively) has a line width-dependent detection limit of 4.5×10^{21} spins / T, and requires 75 μM DMPO for detection with an SNR of 10 or greater. High frequency ESR is much more sensitive and is capable of detecting radicals in concentrations as low as 10^{-10} M, providing the radicals stay arround long enough for detection (Halliwell, 1989).

Studies by others (Makino, 1982, Finkelstein, 1995) have indicated that although DMPO-H and DMPO-OH radicals are both formed by the sonication of aqueous solutions of DMPO, the most intense and persistant ESR signals that are observed, result from the DMPO-OH radicals. For this reason, the present study focuses specifically on the detection of DMPO-OH radicals that are formed by sonication of aqueous DMPO solutions. Experimental conditions are chosen to reflect the conditions that have been found by others (Makino, 1982) to result in the optimum yield of DMPO-OH radicals with respect to sonication time, and DMPO concentration.

In addition, a complimentary test is used to varify the generation of ·H and ·OH radicals under the conditions of sonication utilized in the experiments. The test

involves the sonication of a Fricke dosimetric solution. Fricke dosimetric solutions are dilute solutions of ferrous ions, in which the ferrous ions are converted to optically measurable ferric ions, due to the generation of ·H and ·OH radicals from water. Fricke solutions are most commonly used to quantitatively measure x-ray dose rates. However, they are also useful for determining whether aqueous ·H and ·OH radicals are being produced by ultrasound (Attix, 1966, Misik, 1995).

3.2 Background and Theory

3.2.1 *Kinetic implications of detecting ·H and ·OH radicals by ESR.*

The primary radicals formed by the sonication of water are ·H and ·OH, as indicated by Reaction 3-1 (Makino, *et al*, 1982).

$$H_2O \xrightarrow{\text{— ultrasound cavitation}} \cdot H + \cdot OH \qquad \textbf{Reaction 3-1}$$

Competing with the ultrasound generation of radicals, are fast second order processes that remove the radicals from solution at near diffusion controlled rates (Table 3-1), with rate constants on the order of 10^7 to 10^{10} $M^{-1}s^{-1}$ at room temperatures (Tabata, T.; Ito, Y.; Tagawa, S., 1991).

Table 3-1 Typical reactions followed by ·H and ·OH radicals *

Reactions	Rate Constants (L mol^{-1} s^{-1})
$\cdot H + \cdot H \rightarrow H_2$	1.3×10^{10}
$\cdot OH + \cdot OH \rightarrow H_2O_2$	5.0×10^9
$\cdot H + \cdot OH \rightarrow H_2O$	7×10^9
$\cdot OH + H_2 \rightarrow H_2O + \cdot H$	6×10^7
$\cdot OH + H_2O_2 \rightarrow H_2O + HO_2$	2.3×10^7
$\cdot H + H_2O_2 \rightarrow H_2O + \cdot OH$	9×10^7

* Rate constants are reported at 25°C.

For the reaction: $\cdot H + \cdot OH \rightarrow H_2O$, the theoretical rate law is:

$$\text{rate} = -d[\cdot H]/dt = -d[\cdot OH]/dt = d[H_2O]/dt = k[\cdot H][\cdot OH] \qquad \textbf{Equation 3-1}$$

For the reaction: $\cdot H + \cdot H \rightarrow H_2$, the theoretical rate law is:

$$\text{rate} = -d[\cdot H]/dt = d[H_2]/dt = k[\cdot H]^2 \qquad \textbf{Equation 3-2}$$

For the reaction: $\cdot OH + \cdot OH \rightarrow H_2O_2$, the theoretical rate law is:

$$\text{rate} = -d[\cdot OH]/dt = d[H_2O_2]/dt = k[\cdot OH]^2 \qquad \textbf{Equation 3-3}$$

Thus, the instantaneous rates of consumption of the radicals are expected to be in approximate proportion to the squares of the instantaneous concentrations of the radicals.

At the onset of the ultrasound generation of radicals, the concentration of radicals is small, and the competing rate of consumption of the radicals is small enough to allow the net yield of radicals to increase in concentration. With continued sonication, the concentration of radicals will continue to increase, causing the rate at which the radicals are consumed to increase. When the rate of consumption of a particular radical equals the rate of production, a steady-state concentration of the radical results, as indicated by Equation 3-4.

The condition for steady-state concentration of radicals is given by:

rate of radical generation = rate of radical consumption **Equation 3-4**

Aqueous ·H radicals have a g factor of 2.002, a hyperfine splitting constant of 50.69 mT, and a line width of 6.67 x 10^{-7} T (Wertz and Bolton, 1972). The LFESR spectrometer being used in this study has a line width dependent sensitivity limit of 4.5 x 10^{21} spins/T. This means that for LFESR detection of ·H radicals to occur, there must be at least 3.0 x 10^{15} spins (4.5 x 10^{21} spins/T · 6.67 x 10^{-7} T) for each line in the 2 line spectrum of ·H. In other words, a steady-state concentration of at least 0.66 μM must exist in the 15 mL sample volume as the scan is performed. By Equation 3-4, in order to achieve a steady-state concentration of 0.66 μM, radicals must be generated from the sonication of the water sample at a rate of approximately 4.4 x 10^{-4} mol·$L^{-1}s^{-1}$ (where the $d[\cdot H]/dt \approx k[\cdot H]^2$). This *may* be accomplished by incorporating an ultrasound source into the sample probe assembly of the spectrometer, and sonicating the sample continuously, as the LFESR scan is performed.

An additional factor to be considered is the theoretical ESR field positions of the ·H radical at the 250 MHz operating frequency of the LFESR spectrometer. At 250 MHz, only one ESR transition is possible, and would be observed at a field position of approximately 16 mT. This field position corresponds to the upper limit of the 0 to 16 mT magnetic field sweep range LFESR spectrometer, further

compromising the direct detection of ·H radicals by means of LFESR spectroscopy (See Appendix D for the calculations of energy levels of the ·H radical, and the ESR field positions at 250 MHz.)

Note: Aqueous ·OH radicals have never been detected by ESR. Besides being too short lived, the radicals have a very broad theoretical line width due to a high degree of spin-orbit coupling. Any signal resulting from ·OH radicals would be washed out across the broad line, making detection highly unlikely (Sargent, et al, 1976).

3.2.2 *Trapping ·H and ·OH radicals with DMPO.*

An approach to generate sufficiently high steady-state concentrations of the short-lived ·H and ·OH radicals, is to react the short lived radicals with a spin trap. This converts the short lived radicals to longer lived radicals (spin trap adducts), that will reach stable concentrations that may be within the detection limit of the spectrometer. This eliminates the requirement of continuous sonication during LFESR scans, and also allows lower intensity ultrasound sources to be utilized. The spin trap of choice for the study of aqueous ·H and ·OH radicals has been 5,5-dimethyl-1-pyrroline N-oxide, DMPO (Finkelstein, 1980, Christman, 1987, Misik, 1995).

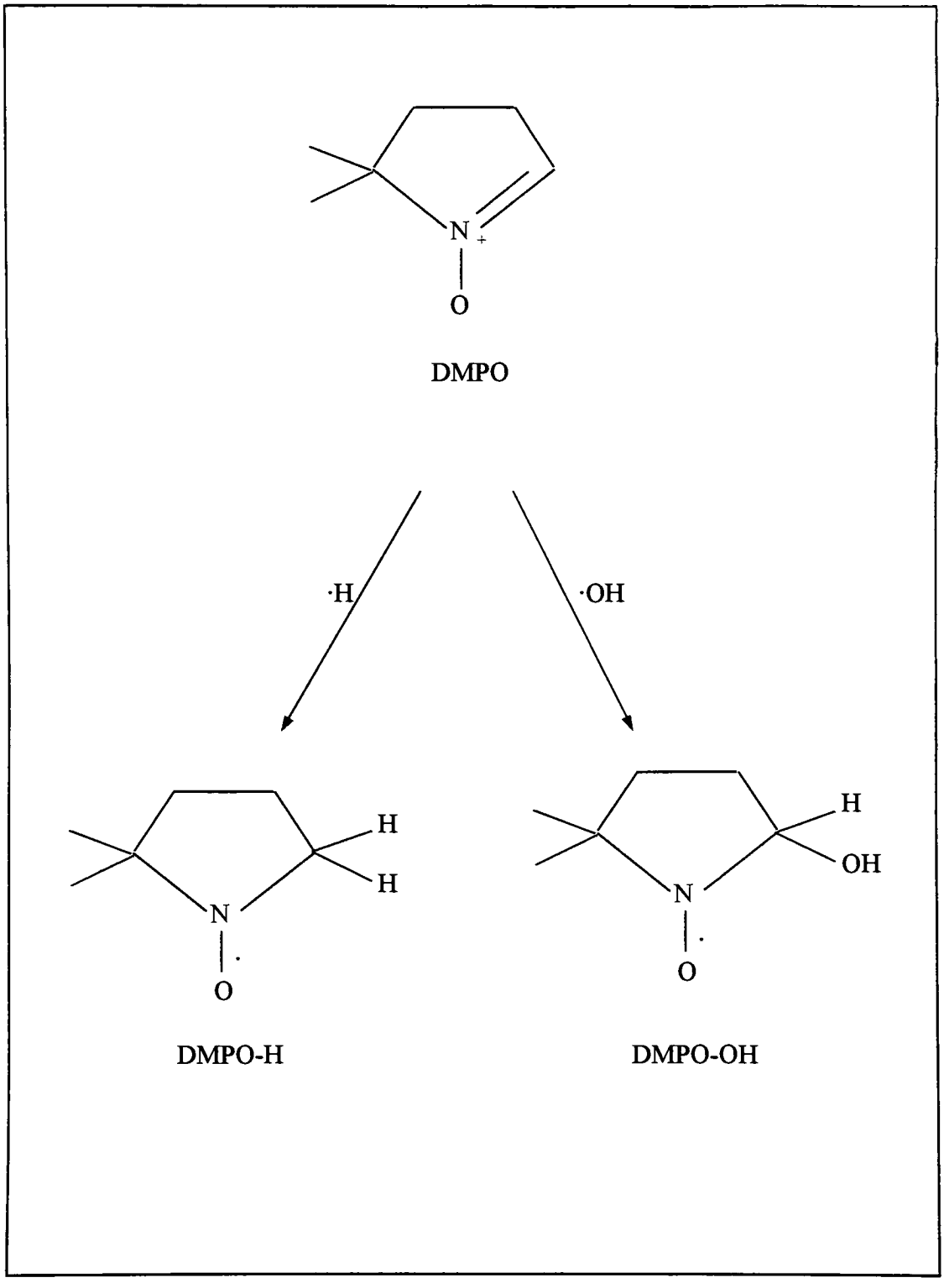


Figure 3-1 Structures of the DMPO spin trap, and the DMPO-OH and DMPO-H adducts.

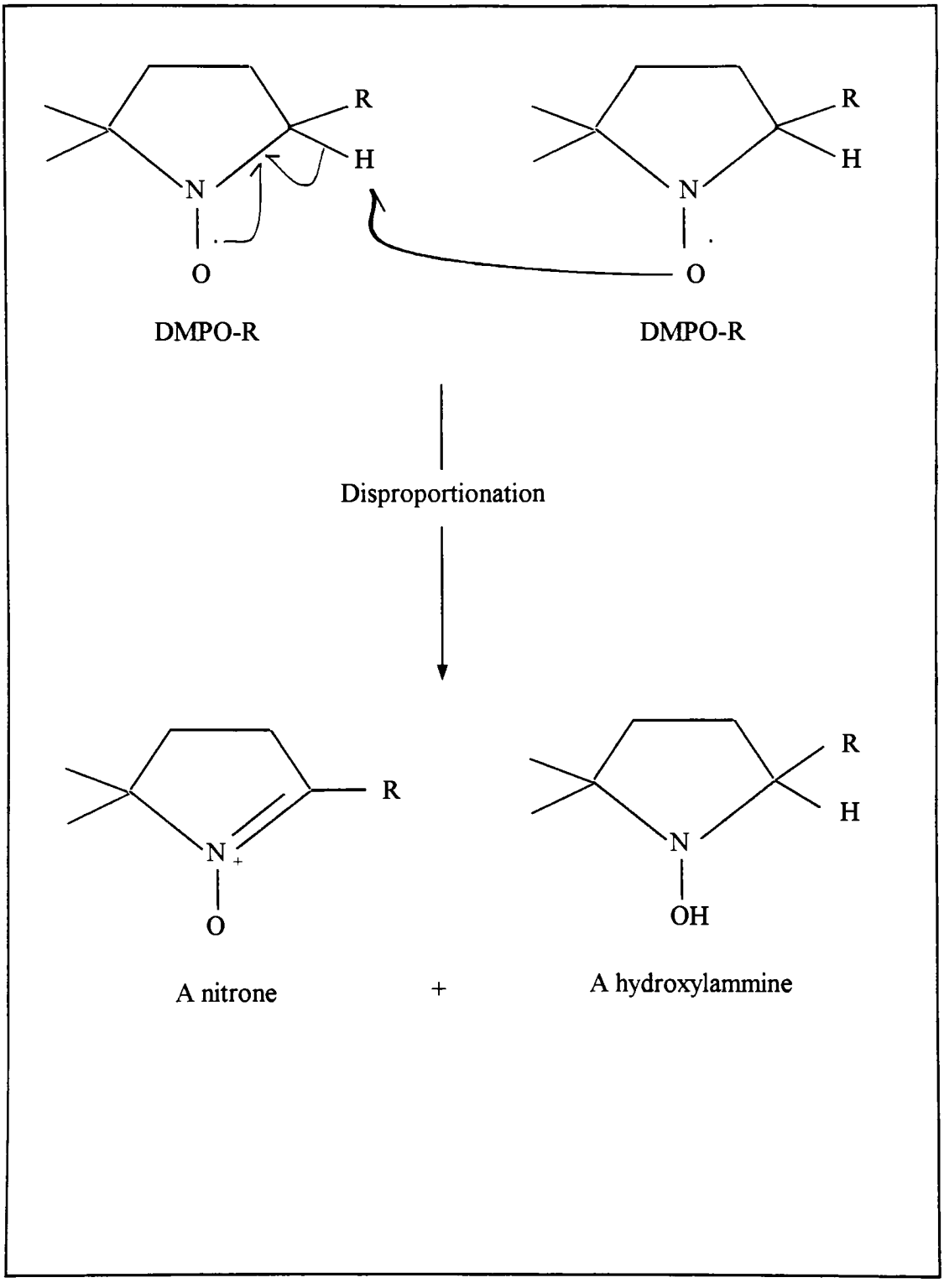
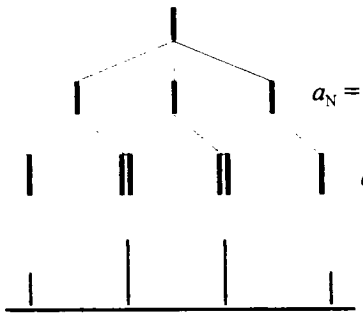
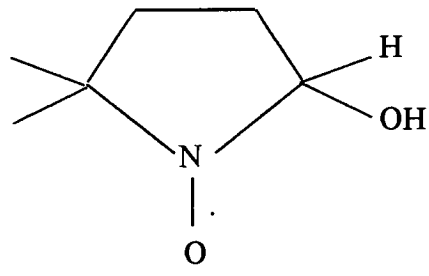


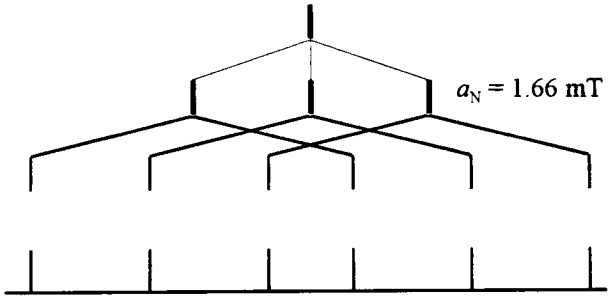
Figure 3-2 Disproportionation of DMPO-R adducts, where R can be H or OH, as in the DMPO-H and DMPO-OH adducts.



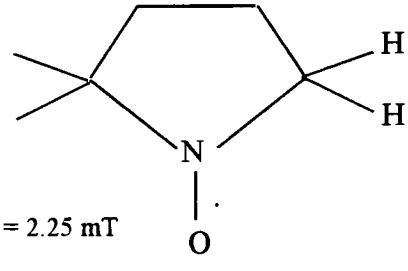
Relative peak intensities = 1 : 2 : 1



DMPO-OH



Relative peak intensities = 1 : 1 : 1 : 1 : 1 : 1



DMPO-H

Figure 3-3 ESR stick structures for the DMPO-H, DMPO-OH, and combined DMPO-H and DMPO-OH adducts. Relative peak intensities are shown.

DMPO falls into the category of nitrone spine traps (Berliner, 1979). When ·H and ·OH radicals are formed in the presence of DMPO, a fraction of the ·H and ·OH radicals will undergo addition to the DMPO to form DMPO-H and DMPO-OH adducts (Reactions 3-2 and 3-3) The structures of DMPO, DMPO-H and DMPO-OH are given in Figure 3-1.

$$\underset{\text{(nitrone spin trap)}}{\text{DMPO}} + \underset{\text{(radical)}}{\cdot H} \rightarrow \underset{\text{(nitroxide adduct)}}{\text{DMPO-H}} \qquad \textbf{Reaction 3-2}$$

$$\text{DMPO} + \cdot OH \rightarrow \text{DMPO-OH} \qquad \textbf{Reaction 3-3}$$

The reactions of DMPO with ·H and ·OH follow second order kinetics. The rate of formation of adducts is directly proportional to the concentration of DMPO as well as the radical that is being trapped. The theoretical second order rate laws for reactions 3-2 and 3-3 are given in Equations 3-5 and 3-6, respectively.

Equation 3-5

$$\text{rate} = -d[\text{DMPO}] / dt = -d[\cdot H] / dt = d[\text{DMPO-H}] / dt = k_{\text{DMPO+·H}}[\text{DMPO}][\cdot H]$$

Equation 3-6

$$\text{rate} = -d[\text{DMPO}] / dt = -d[\cdot OH] / dt = d[\text{DMPO-OH}] / dt = k_{\text{DMPO+·OH}}[\text{DMPO}][\cdot OH]$$

The rate constant for the formation of DMPO-OH, is reported to be 4.3×10^9 $M^{-1}s^{-1}$. The rate constant for the formation of DMPO-H is not reported in the literature, but is generally regarded to be of the same order of magnitude as the rate

constant for the formation of DMPO-OH (Finkelstein, 1980b).

The fraction of ·H and ·OH that is successfully trapped by DMPO during a given experimental time frame, depends on how well the DMPO will compete against the fast reactions that normally remove the ·H and ·OH from the system. The second order rate constant of 4.3×10^9 $M^{-1}s^{-1}$ implies successful competition for the ·H and ·OH radicals, by the DMPO, when compared to the second order rate constants for the recombination of the radicals (Table 3-1). However, for the reaction of DMPO with ·H and ·OH to result in sufficient concentrations of DMPO-H or DMPO-OH adducts, the adducts themselves must be sufficiently persistent. In other words, the DMPO is not just competing with mechanisms that remove the un-trapped radicals from solution, but also with any mechanisms that removes the *trapped* radicals from the solution. In other words, the stability of the trap adducts must be considered.

3.2.3 *Stability of the DMPO-H and DMPO-OH adducts.*

The DMPO-H and DMPO-OH adducts are nitroxide radicals. Nitroxides are π-radicals, with the unpaired electron shared between the N and the O in a π^* orbital. The result of the delocalization of the unpaired electron is a relatively stable radical. However, the stabilities of different nitroxides vary significantly with respect to the degree of substitution at the beta carbons (i.e. the carbons that are directly bonded to the nitroxide function). Nitroxides that are used as spin *labels* (i.e. nitroxide radicals

that used to synthetically label diamagnetic compounds) typically have tert-alkyl functions at the beta position of the nitroxide. This form of a nitroxide is usually very stable, and persists for days or weeks at room temperature, with no significant loss of the nitroxide label through subsequent reactions. However, the nitroxides that are used for spin *trapping* are less substituted, having at least one hydrogen at the beta position to the nitroxide function. The hyperfine splitting that results from the alpha protons is highly sensitive to the radical that is added. This is an advantage in spin trapping techniques, where identification of various trapped radicals is made possible through analysis of the distinct hyperfine splitting patterns that result from the beta proton. However, the presence of one or more protons at the beta position destabilizes the nitroxide, by making the nitroxide susceptible to disproportionation. Disproportionation is the bimolecular process in which one mole of nitroxide deprotonates the beta carbon of a second mole of the nitroxide, leading to the formation of one mole of the nitrone and one mole of a hydroxylammine compound (See Figure 3-2).

The DMPO-H and DMPO-OH adducts that are formed by the sonolysis of aqueous spin trap solutions are both known to decay with second order as well as first order rate components (Finkelstein, 1980, Misik, 1995). At high concentrations of spin adducts, the second order process of disproportionation dominates. At low concentrations of the adducts, the first order process of direct oxidation of the

nitroxide function dominates the rate of decay. The oxidation of the nitroxide function is caused by the presence of H_2O_2 that is formed by the recombination of hydroxyl radicals. Small increases in the concentration of H_2O_2 cause the first order half life of DMPO-OH to decrease significantly.

Although the rate constants and half lives of the decay processes are not well known, it is well understood that the nitroxide adducts that are formed by 50 kHz sonolysis have stable concentrations that plateau at approximately 0.67 μM (Makino, 1982). This is the concentration that exists for at least several minutes long without significant decay, and is therefore considered to be stable for the time frame of a typical ESR scan. Also, the DMPO-H adducts are much less stable than the DMPO-OH adducts. This is probably due to the fact that DMPO-H adducts have two beta hydrogens, significantly increasing the probability of disproportionation. The spectra that are observed show relatively weak intensities for DMPO-H, as compared to DMPO-OH (Misik, 1995).

3.2.4 *Minimum detectable concentration of DMPO-OH adduct for LFESR detection.*

The stick representations of the DMPO-H and DMPO-OH ESR spectra are given in Figure 3-3. The spectrum of the DMPO-H adduct has hyperfine splitting constants of a_N = 1.66 mT and a_H^β = 2.25 mT. The spectrum of the DMPO-OH adduct has hyperfine splitting constants of $a_N = a_H^\beta$ = 1.49 mT. The line widths of the

spectra are approximately 1.2 x 10^{-4} T at 9.5 GHz (Makino, *et al*, 1982). The line width at 250 MHz is expected to be approximately 2.5 x 10^{-4} T (i.e. approximately equal to the measured line width of Fremy's salt at 250 MHz). As with the Fremy's salt nitroxide standard used in Part II, there will be a slight asymmetry in the hyperfine splitting of the DMPO-OH radicals at 250 MHz. This is due to the non-linear divergence of energy levels at low frequencies (See Appendix D).

The DMPO-OH adduct concentration is expected to be higher, giving rise to more intense lines (Misik, 1995). The most intense lines of the DMPO-OH spectrum are the second and third lines. These lines each represent 33% of the total number of spins contributing to the spectrum (See Figure 3-2). The line width dependent detection limit of the LFESR spectrometer was determined earlier to be 4.5 x 10^{21} spins T^{-1} Under optimum conditions of sensitivity, field-modulation is expected to double the line width of DMPO-OH from 2.5 x 10^{-5} T to 5.0 x 10^{-5} T. The minimum detectable concentration of DMPO-OH is calculated to be:

$$\frac{4.5 \times 10^{21} \text{ spins}}{\text{T}} \quad \frac{5.0 \times 10^{-5} \text{ T}}{} \quad \frac{1}{1/3} \quad \frac{1 \text{ mol}}{6.02 \times 10^{23} \text{ spins}} \quad \frac{1}{0.015 \text{ L}} = 75 \ \mu\text{M}$$

Where, 1/3 is the statistical weight of the most intense line in the spectrum.

3.2.5 *Verification of ·H and ·OH production by Fricke dosimetry.*

Fricke dosimetry (Attix, 1966, Misik, 1995) can be used to verify the production of ·H and ·OH radicals by ultrasound. A Fricke dosimetric solution is an aerated solution of 0.001 M ferrous sulfate in 0.8 N sulfuric acid. The production of ·H and ·OH radicals, as well as H_2O_2 in the Fricke dosimetric solution causes Fe^{2+} ions to be oxidized to Fe^{3+} ions. The oxidation reactions are:

$$Fe^{2+} + \cdot OH \rightarrow Fe^{3+} + OH^- \qquad \text{Reaction 3-4}$$

$$\cdot H + O_2 \rightarrow HO_2 \qquad \text{Reaction 3-5}$$

$$Fe^{2+} + HO_2 \rightarrow Fe^{3+} + HO_2^- \qquad \text{Reaction 3-6}$$

$$HO_2 + H^+ \rightarrow H_2O_2 \qquad \text{Reaction 3-7}$$

$$Fe^{2+} + H_2O_2 \rightarrow Fe^{3+} + \cdot OH + OH^- \qquad \text{Reaction 3-8}$$

Reactions 3-4 through 3-7 are fast, and compete well with the radical-radical recombination reactions listed in Table 3-1. Reaction 3-8 proceeds slower, with a rate constant on the order of 56 $M^{-1}s^{-1}$.(Allen, 1961).

The concentration of Fe^{3+} in Fricke dosimetric solutions, is typically determined by measuring optical absorbance at 224 nm. At 224 nm and 25 °C, Fe^{3+}

ions have a molar extinction coefficient, ε, of 4565 $M^{-1}cm^{-1}$ (Attix, 1966).

In the absence of ultrasound, the Fricke solution will slowly oxidize at a rate that is proportional to the square of the ferrous ion concentration and the first power of oxygen concentration. In a 0.01 M solution, the concentration of ferrous ion decays at a rate of 2×10^{-6} M / day at 25 °C. At a concentration of 0.001 M Fe^{2+}, the solution is quite stable, when kept refrigerated. The slight increases in optical absorbance at 225 nm due to slow decay of the unsonicated solution, is corrected for by using the unsonicated solution as the reference solution in any optical measurements.

3.3 Experimental Methods

After characterizing the performance of the LFESR spectrometer, a series of preliminary experiments were conducted to test the effectiveness of the spectrometer in the detection of DMPO-OH spin trap adducts, formed by a 20 kHz ultrasound source.

Each of the experiments performed here utilized a 20 kHz, 0-120 watts / cm^2 Vibracell ultrasonic transducer (Model # VC50, Sonic & Materials, Inc., Danburg, CT, USA). Ultrasound power levels are based on test data supplied by the manufacturer. The spin trap solutions were prepared from 5,5-dimethyl-1-pyroline N-oxide (catalogue # D-5766, Sigma Chemical Company, St. Louis, MO, USA) and HPLC grade water. DMPO has a formula weight of 113.2 and a density of 1.02 g/ml at 25 °C. In each case, the sample solution was prepared by dilution of a stock DMPO solution. The stock DMPO solution was prepared to have a concentration of 10 mM. Samples were sonicated in 16 x 150 mm disposable culture tubes, under various conditions as specified in each of the following descriptions. Sample sonication was performed using the set-up illustrated in Figure 3-4.

LFESR scans were made with the modulation power set at 1 watt / 8 ohms, the RF power attenuation set at 10 dB, and a pre-time constant of 300 ms. The level of signal averaging was set as indicated in each procedure.

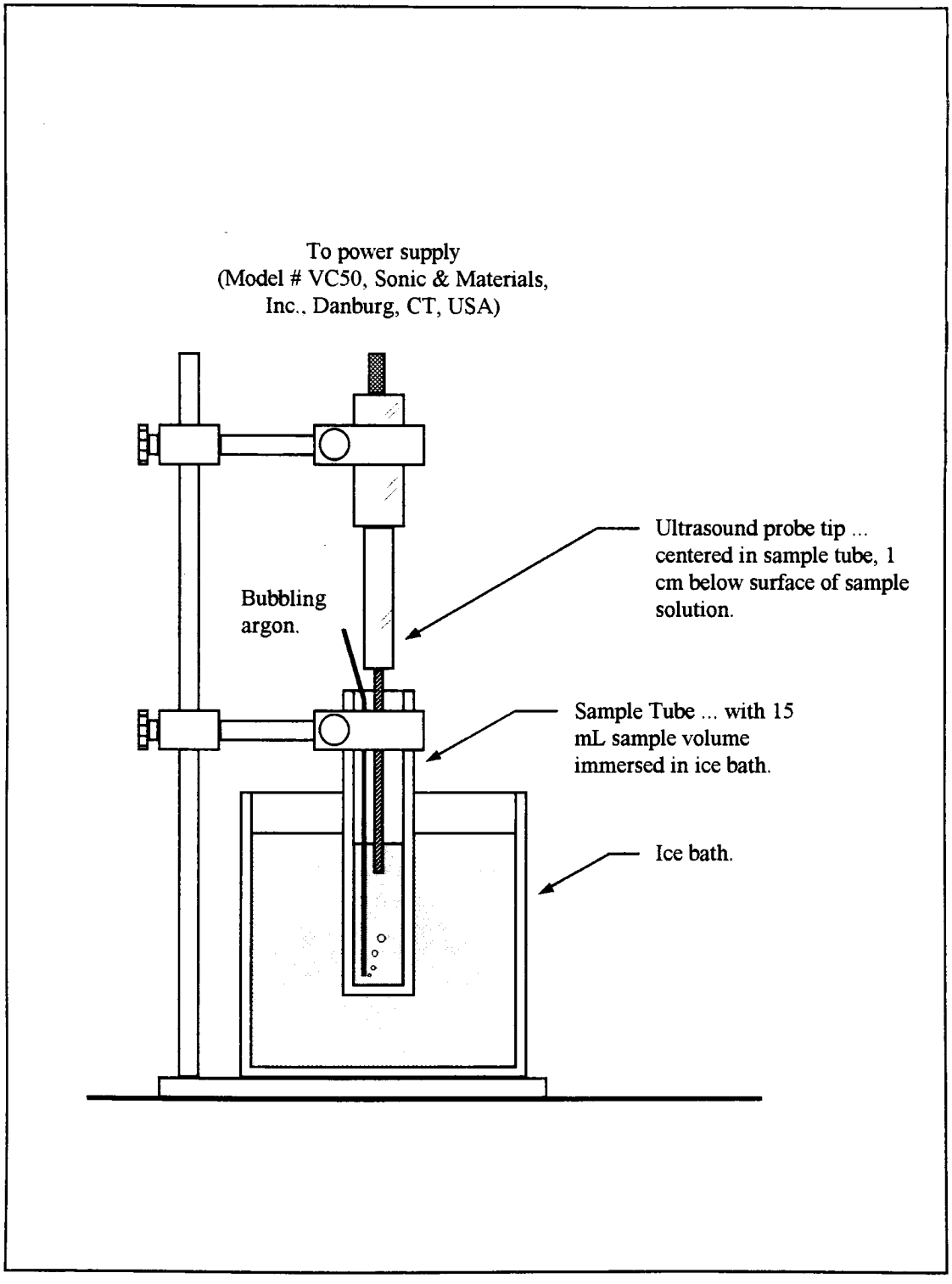


Figure 3-4 20 kHz sonication apparatus. This apparatus is used for the sonication of DMPO and Fricke solutions under various experimental conditions. No argon is bubbled during sonication of Fricke solutions.

3.3.1 *Sonication of Fricke Solution.*

In this procedure, a 15 mL Fricke dosimetric solution was sonicated at full output (60 watts / cm^2) for a period of 15 minutes, using the set-up shown in Figure 3-3. The tip of the ultrasound transducer was positioned 1cm under the surface of the DMPO solution, and centered with respect to the sides of the sample tube. To prevent bulk heating of the solution, the sample tube was positioned in an ice-water bath during the sonication process. Before sonication, the stock Fricke solution was vigorously shaken to provide aeration of the solution. After sonication, a 3 mL aliquot of the sonicated Fricke solution was transfered to a quartz UV-vis spectrometer cuvete for analysis of optical absorbance at 224 nm. Unsonicated Fricke solution was used as the reference. This procedure was repeated for sonication times of: 30, 45, and 60, 75, 90, 105 and 120 minutes. The procedure was performed a total of 4 runs for each sonication time. The entire procedure was repeated at a sonication intensity of 120 watts / cm^2. A plot of absorbance vs. sonication time was made for each intensity level (See Figure 3-5).

3.3.2 *Reference scan of DMPO.*

In this procedure, a reference scan of un-sonicated 1 mM DMPO sample solution was obtained. The sample solution was scanned from 6 to 12 mT in steps of 0.005 mT, with signal averaging set at n = 4. This scan range is chosen to bracket the

various LFESR peaks that could arise from ESR absorption by DMPO-H or DMPO-OH adducts. The sample was scanned at the equilibrium operating temperature of the LFESR spectrometer sample probe, which was approximately 28 °C (See Figure 3-6).

3.3.3 *Sonication and LFESR scan of 1 mM DMPO solution - (20 minute scan).*

In this procedure, a 15 ml, 1 mM aqueous solution of DMPO was sonicated at full output (120 watts / cm^{-2}) for a period of 15 minutes, using the set-up shown in Figure 3-4. The tip of the ultrasound transducer was positioned 1cm under the surface of the DMPO solution, and centered with respect to the sides of the sample tube. To prevent bulk heating of the solution, the sample tube was positioned in an ice-water bath during the sonication process. Argon was continuously bubbled through the solution at a rate of ~ 50 mL / minute for 20 minutes before sonication, and for the 15 minute sonication period. By saturating the solution with argon, the thermal conductivity of the solution is lowered, slowing the rate of decay of DMPO-H and DMPO-OH (Makino, 1982). At the end of the sonication process, the sonicated sample was immediately transferred to the STS sample probe, and scanned from 6 to 12 mT, in steps of 0.005 mT. Signal averaging was set at a level of n = 4 for this scan. The time that elapsed between the end of the sonication processed the start of the scan was approximately 40 seconds. The time required for the scan was 20 minutes (See Figure 3-7).

3.3.4 *Sonication and LFESR scan of 1 mM DMPO solution - (4 minute scan).*

The same procedure was followed as described in 3.3.3., except the LFESR scan was performed from 9.2 to 9.8 mT, in steps of 0.01 mT. The shortened scan range is chosen to bracket the maximum expected spectral peaks that could result from either the combined absorption from DMPO-H and DMPO-OH, or from the exclusive absorption from either DMPO-H or DMPO-OH. By narrowly focusing the scan range, as well as reducing the resolution to 0.01 mT, the required scan time was reduced to a period of 4 minutes, thus increasing the likelihood of detecting the signals of a possibly unstable spin adduct (See Figure 3-8).

3.4 Results and Discussion

3.4.1 *Sonication of Fricke solution.*

The absorbance of the Fricke solution was found to increase in a linear relationship with increased sonication time, as shown in Figure 3-5. At a sonication level of 60 watts / cm^2, the slope of the plot of $[Fe^{3+}]$ vs. sonication time is 1.77×10^{-7} M $min.^{-1}$ At a sonication level of 120 watts / cm^2, the slope was 3.2×10^{-7} M. The ferric concentration remained stable over a period of 24 hours following the sonication procedures, with the sample refrigerated. The results indicate that the sonication conditions are resulting in the formation of radicals, and that the radical yield is in linear proportion to the sonication time. Also, an increase in sonication intensity results in the formation of radicals at a higher rate. However, these results do not necessarily reflect the concentration of DMPO trap adducts that should be formed under the same conditions, since the DMPO adducts that are formed are expected to undergo disproportionation and oxidation reactions, with second and first order rates, respectively.

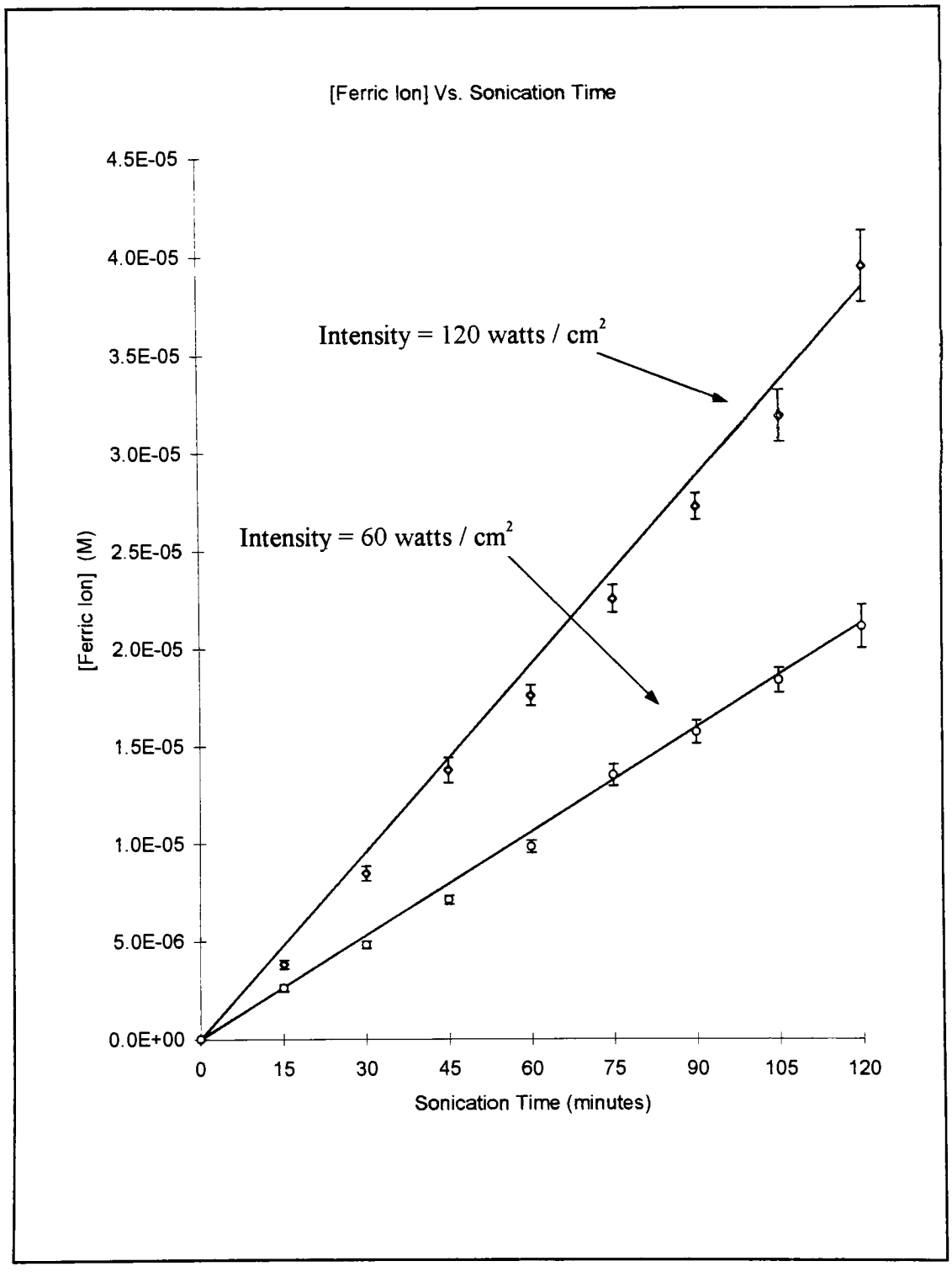


Figure 3-5 Sonication of Fricke solution. A plot of ferric ion concentration vs. sonication time.

3.4.2 *Reference scan of DMPO.*

Figure 3-6 represent the LFESR spectrum of an unsonicated 1 mM aqueous solution of DMPO. The 6 to 12 mT scan range is sufficient to bracket all peaks that may occur due to either or both DMPO-H and DMPO-OH adducts. The peak-to-peak activity of the spectrum is interpreted here as noise. The standard deviation, s, of this activity is measured to be 2.89×10^{-8}, which is the same magnitude as for a water sample with no DMPO. This scan will be used as the reference that sonicated DMPO samples will be compared to.

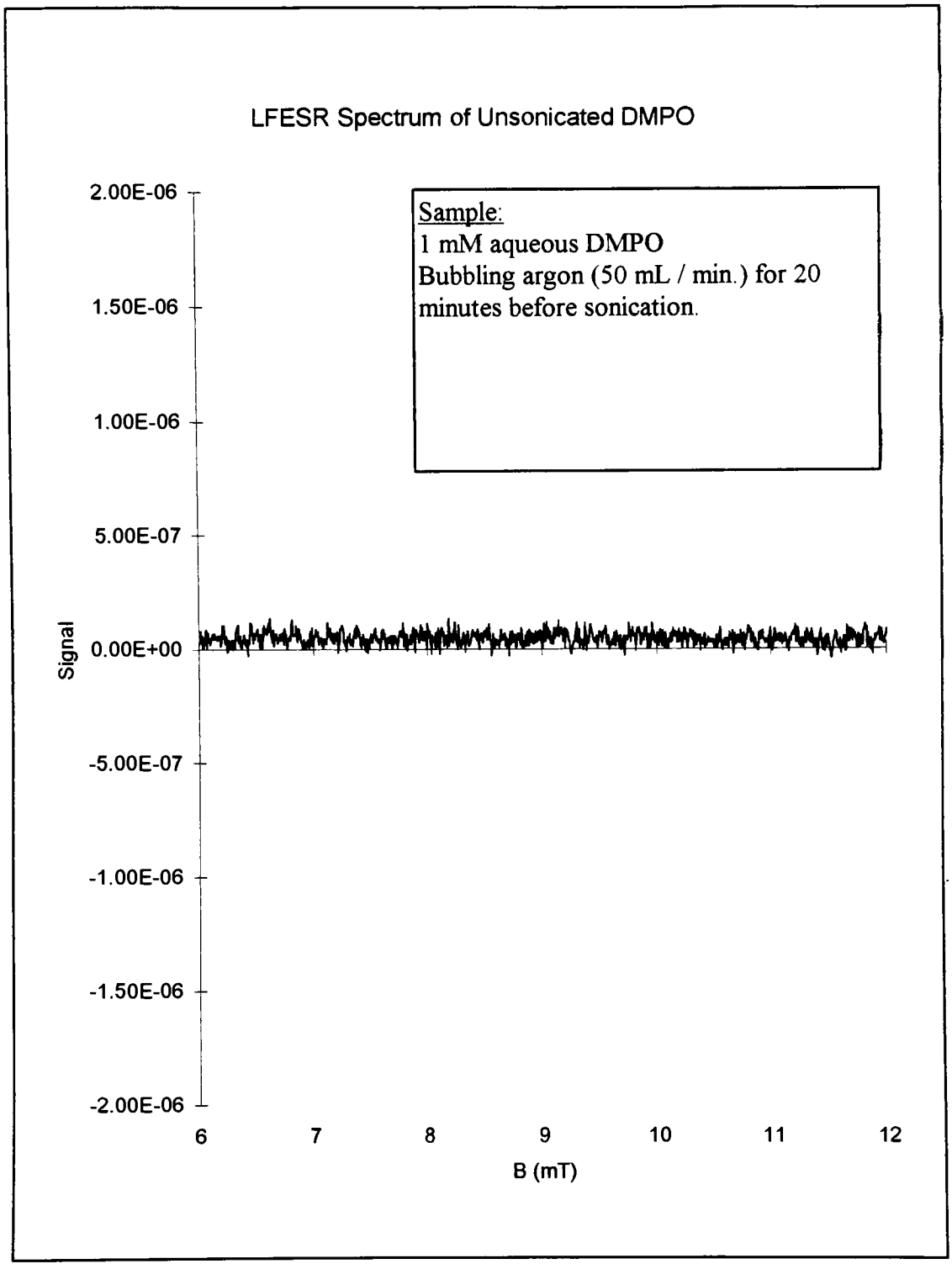


Figure 3-6 A reference LFESR scan of an unsonicated 1 mM aqueous solution of DMPO.

3.4.3 *Sonication and LFESR scan of 1 mM DMPO solution - (20 minute scan).*

Figure 3-7 represents an LFESR scan of a 1 mM aqueous solution of DMPO, sonicated for 15 minutes at room temperature. No apparent absorption peaks are observable in this scan. DMPO-H and DMPO-OH peaks have true line widths of approximately 0.025 mT. The modulation power setting being used here (1 watt / 8 ohms) corresponds to optimized peak-to-peak amplitude, as well as an approximate doubling of the line width, to a value of 0.05 mT. Thus any peaks that may result from the presence of DMPO-OH or DMPO-H radicals have the greatest chance of being detected at this setting of modulation power, and should be resolvable by the field sweep step size of 0.005 mT. This result could be interpreted to mean several different things. First, it is possible that the conditions of sonication did not lead to the generation of ·H and ·OH radicals. Second, it is possible that the radicals were generated, but were not effectively trapped by the DMPO. Third, it is possible that the radicals were generated, and trapped, but that the concentration of spin trap was not within the 75 μM detection limit of the LFESR spectrometer.

Of the three possible explanations mentioned here, the third seems most valid. The first possibility can probably be dismissed, based on the results of sonicating the Fricke solution, which demonstrated the ultrasound generation of free radicals, under nearly equivalent but not identical conditions. The difference in the two procedures was that the Fricke solution was sonicated without being bubbled with argon, while

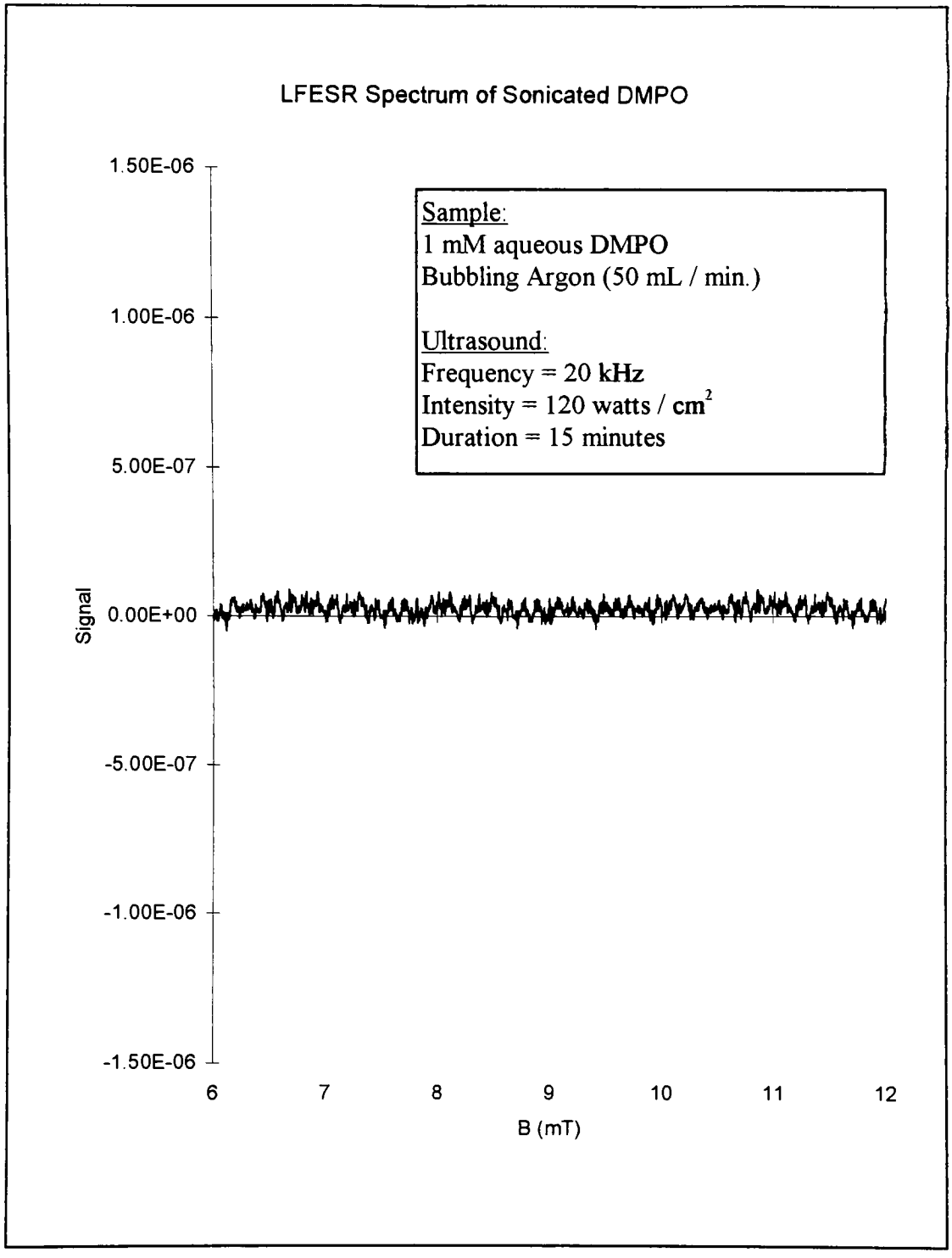


Figure 3-7 An LFESR scan of a 1 mM aqueous solution of DMPO, sonicated for 15 minutes.

the DMPO solution was sonicated with bubbling argon. This should not be a concern, since bubbling argon gas has been shown to enhance the ultrasound generation of radicals (Mckee, 1977). The second possibility of generating but not trapping the radicals is not entirely likely, since DMPO is known to react rapidly with ·H and ·OH radicals to form DMPO-H and DMPO-OH, and has been successfully used to trap the radicals in similar studies, under similar conditions. The third possibility that the concentration of trap adducts was not within the 75 μM detection limit of the spectrometer seems most likely. The most compelling evidence in favor of this possibility is that the highest ESR detected concentration of DMPO-OH that has been reported in the literature is 0.67 μM, and that this figure corresponds to a maximum yield with respect to optimization of sonication time, and DMPO concentration (Makino, 1982). Furthermore, these optimized conditions were reflected in the procedures followed in this study.

It is expected that the concentration of DMPO-spin adducts is at a maximum at the completion of the sonication process and that it immediately begins to decay over time from the instant that the sonication is removed from the sample solution. The decay rate should be the highest at the instant the sonication is removed, since the second order process of disproportionation is proportional to the square of the concentration of DMPO-H and DMPO-OH adducts in the solution. Also, the first order oxidation of the trap adducts by H_2O_2, will be at a maximum, since the

concentration of H_2O_2 will be at a maximum at the instant the sonication process is stopped. As the H_2O_2 is consumed by the oxidation of spin trap adducts, the rate of oxidation will decrease linearly. With this in mind, it is possible that the initial concentration of trap adducts immediately after sonication may exceed the 75 μM detection limit of the spectrometer. (As mentioned earlier, it is possible that the sonication process is generating more spin trap adducts than indicated by the Fricke tests, since the trap mechanism may be more efficient than the Fricke mechanism.) It may be possible to detect a signal if the scan is acquired within a shorter period of elapsed time after the sonication of the sample. This possibility was tested in 3.3.4.

3.4.4 *Sonication and LFESR scan of 1 mM DMPO solution - (4 minute scan).*

Figure 3-8 represents an LFESR scan of a 15 mL, 1 mM aqueous solution of DMPO, sonicated for a period of 15 minutes. This scan was made over a shortened range of 9.2 to 9.8 mT, in steps of 0.005 mT, which meant that the time required to acquire the scan was shortened from 20 minutes to 4 minutes, affording a greater possibility of detection. There are no apparent absorption peaks in this scan. A maximum intensity peak is expected to occur at 9.5 mT. This peak would result from either of the two DMPO adducts, or the combination of the two.

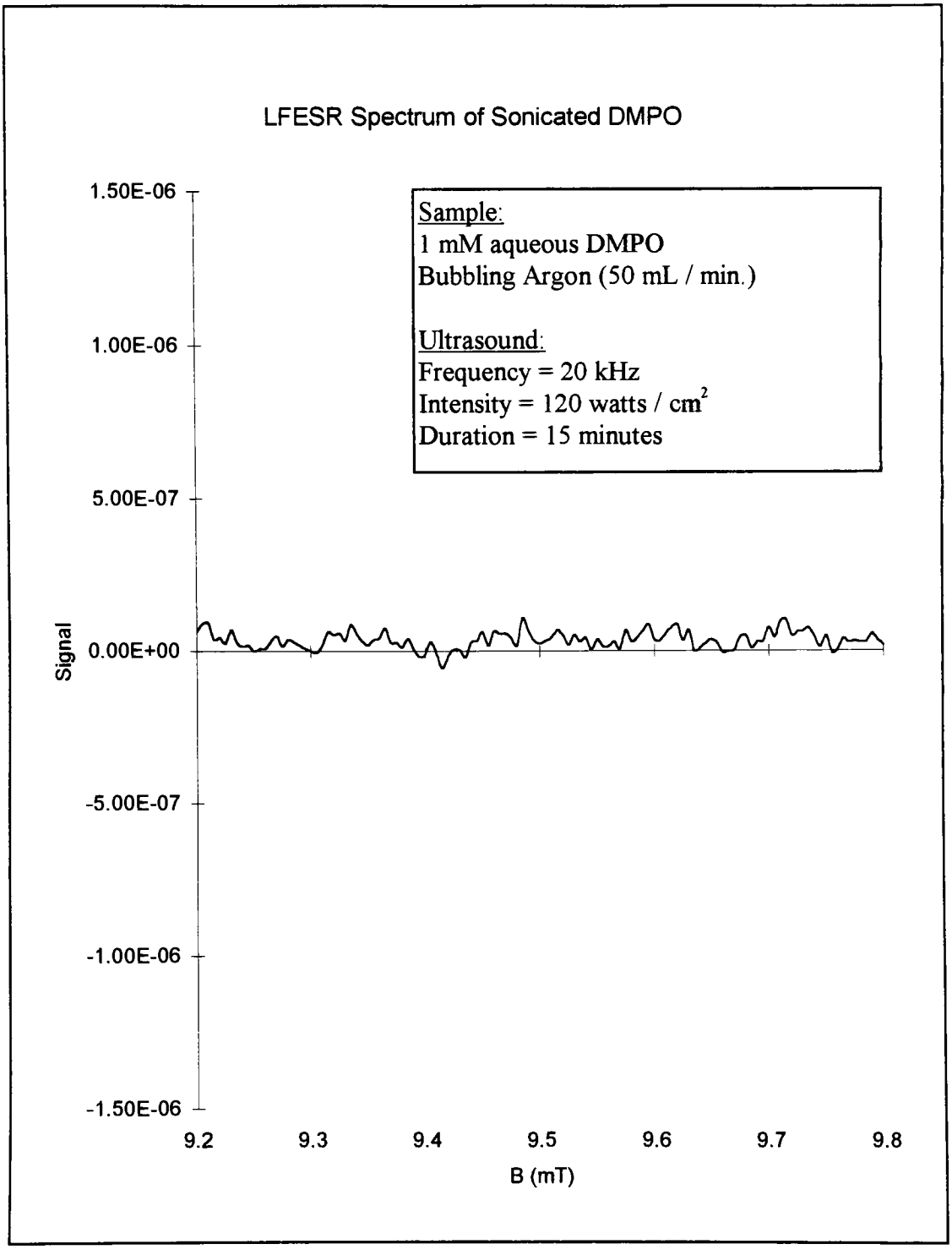


Figure 3-8 An LFESR scan of a 1 mM aqueous solution of DMPO, sonicated for 15 minutes.

3.4 Conclusions

In the current configuration, the LFESR spectrometer has not been demonstrated to be effective in the detection of ultrasound generated DMPO-H or DMPO-OH radicals.

Evidence for the ultrasound generation of ·H and ·OH radicals is found in the results of the Fricke dosimetric tests. The Fricke dosimetric tests show a linear change in optical absorbance at 224 nm, with respect to the duration of sonication. The Fricke tests also show that an increase in ultrasound intensity, from 60 to 120 watts / cm^2, results in an increase in the rate of change in optical absorbance of the Fricke dosimetric solution at 224 nm.

LFESR scans of sonicated aqueous DMPO solutions show that any DMPO-H and DMPO-OH that may be formed during the sonication process, exists in concentrations less than the 75 μM detection limit of the spectrometer at the time that the scan is made. Consideration of the second and first order decay processes experienced by DMPO-H and DMPO-OH adducts, as described by the literature, leads to the conclusion that a lack of spin adduct *persistence* may be a critical factor involved in the resulting low concentrations of DMPO-H and DMPO-OH at the time that the spectral scan is made.

Consideration should be given to using a nitrone spin trap that is more stable through an increased degree of substitution at the beta position of the nitrone. While this would lead to a loss of the informative beta-hyperfine splitting that is useful in identifying the radicals being trapped, the sacrifice may be paid for in the ability to detect the *presence*, if not the *identity*, of radicals generated by ultrasound.

Another approach for future study in this area, is to integrate the ultrasound source with the sample probe of the spectrometer, so that steady state concentrations of DMPO-H and DMPO-OH adducts may be generated by the ultrasound source, as the LFESR scans are being performed. It may be possible to maintain the steady state concentrations at levels that are within the 75 μM detection limit of the spectrometer. This approach addresses the problem that seems to have stood in the way of success in the present study, which is the competitive concentration dependent disproportionation of DMPO-H and DMPO-OH, to result in rapid signal decay when the ultrasound source is removed from the sample.

References:

Allen, A. O.; *The Radiation Chemistry of Water and Aqueous Solutions*; D. Van Norstrand Company, Inc.: New York, 1961.

Attix, F H.; Roesch, W. C.; *Radiation Dosimetery II*; Academic Press: New York, 1966.

Berliner, L. J.; *Spin Labeling Theory and Applications II;* Academic Press: New York, 1979.

Christman, C. L., Carmichael, M. M., Riesz, P "Evidence for free radicals produced in aqueous solutions by diagnostic ultrasound."; *Ultrasonics* **1987**, 25, 31-34.

Finkelstein, E.; Rosen, G. M.; Raukman, E. J. "Spin Trapping of Superoxide and Hydroxyl Radical: Practical Aspects"; *Arch. Biochem. Biophys.* **1980**, 200, 1-16.

Fleischer, A. C.; James, A. E., *Diagnostic Sonography: Principles and Applications*; W. B. Saunders Company, 1989.

Halliwell, B.; Gutteridge, J. C., *Free Radicals in Biology and Medicine*; Oxford University Press: New York, 1989.

Halpern, H. J., Spencer, D. P., von Polen, J.; Bowman, M. K.; Nelson, A. C.; Dowey, E. M.; Teicher, B. A. "Imaging radio frequency electron-spin-resonance and sensitivity for in vivo measurements"; *Rev. Sci. Instrum.* **1989,** 60, 1040-1050.

Hornak, J.P., Spacher, M.; Bryant, R.G. "A modular low frequency ESR spectrometer"; *Meas. Sci. Technol.* **1991,** 2, 520-522.

Jones, M. T.; "Electron spin exchange in aqueous solutions of $K_2(SO_3)_2NO$", *J. Chem. Phys.* **1963,** 38(12), 2892-2895.

Knapp, R. T.; Daily, J. W.; Hammitt, F. G.; *Cavitation*; McGraw-Hill, New York, 1970.

Lurie, D. J.; Bussell, D. M.; Bell, L. H.; Mallard, J. R. "Proton-electron double magnetic resonance imaging of free radical solutions"; *J. Magn. Reson.* **1988**, 76, 366-370.

Makino, K.; Mossoba, M. M.; Riesz, P. "Chemical Effects of Ultrasound on Aqueous Solutions. Formation of Hydroxyl Radicals and Hydrogen Atoms."; *J. Phys. Chem.* **1982**, 87, 1369-1367.

Misik, V.; Miyoshi N.; Riesz P "EPR Spin-Trapping Study of the Sonolysis of H2O/D2O Mixtures: Probing the Temperatures of Cavitation Regions."; *J. Phys. Chem.* **1995**, 99, 3605-3611.

Montgomery, C. G.; *Techniques of Microwave Measurements: MIT radiation laboratory series*; McGraw-Hill: New York, 1947; p. 195.

Neta, P.; Steenken, S.; Janzen, E. G.; Shetty, R. V "Pattern of addition of hydroxyl radicals to the spin traps α-pyridyl 1-oxide n-ter-butyl nitrone"; *J. Phys. Chem.* **1980,** 84, 532-534.

Poole, C. P.; Farach, H. A.; *Theory of Magnetic Resonance. Second Edition.*; John Wiley and Sons: New York, 1897

Robertson, J. M.; Wislicenus, G. F.; *Cavitation State of Knowledge*; The American Society of Engineers, United Engineering Center: New York, 1969.

Sargent, E. P.; Gardy, E. M. "Spin trapping of radicals formed during radiolysis of aqueous solutions. Direct electron spin resonance observations."; *Can. J. Chem.* **1976**, 54, 275-279.

Suslick, K.; "Sonochemistry"; *Science* **1990**, 247, 1439-1445.

Swanson, D. P.; Chiton, H. M.; Thrall, J. H.; *Pharmaceuticals in Medical Imaging*; MacMillan Publishing Co., Inc.. New York, 1990.

Szczepaniak, E.; Hornak, J. P. "ESR Imaging Based on the Modulation-Field Phase."; *J. Magn. Reson.* **1993**, 104, 315-320.

Tabata, T.; Ito, Y.; Tagawa, S.; *Handbook of Radiation Chemistry;* CRC Press: Boston, 1991.

Wertz, J.E.; Bolton, J.R.; *Electron Spin Resonance: Elementary Theory and Practical Applications*; McGraw-Hill: New York, 1972.

Appendix A - LFESR Component List

Component	Description, Part Number, and Source
A1	5-500 MHz, 22.5 db Gain Signal Amplifier UTC-517-1 Aventek, Mitpitas, CA, USA
A2	150 kHz - 300 MHz, 10 watt Power Amplifier 411-LA ENI Inc., Rochester, NY, USA
A3	Audio Frequency Power Amplifier M-0.5t Carver, Lynnwood, WA, USA
R1, R2, R4	0-10 db, 1 db-step, Turret Attenuators 5010A Wavetek Corporation, San Diego, CA
R3	0-70 dB, 10 dB-step, Turret Attenuator 5070A Wavetek Corporation, San Diego, CA
B1, B2, B5	2 - 500 MHz RF Power Divider PE 2000 Pasternack Enterprises, Irvine, CA, USA
B3	180 Degree Hybrid Tee (Return Loss Bridge) HJ-300 Merrimac Industries, Inc., West Caldwell, NJ, USA
B4	-20 dB Directional coupler CP-20-215 Merrimac Industries, Inc., West Caldwell, NJ, USA
Modulation Coils and RF Loop	Custom Assembly Fabricated by: Joseph P. Hornak Rochester Institute of Technology, Rochester, NY, USA

LFESR Component List, continued.

Component	Description, Part Number, and Source
M1, M2	Doubly Balanced Mixer DM-2A-200 Merrimac Industries, Inc., West Caldwell, NJ, USA
D1	1 GHz diode detector PE-8000-50 Pasternack Enterprises, Irvine, Ca, USA
F1, F2	124 MHz, LC Bandpass Filter Custom Component Fabricated by Joseph P. Hornak, Rochester Institute of Technology, Rochester, NY, USA
F3	10 kHz, LC Bandpass Filter Custom Component Fabricated by Joseph P. Hornak, Rochester Institute of Technology, Rochester, NY, USA
0-160 MHz RF Source	Binary Coded Decimal (BCD) Controlled, 0 to 160 MHz (±0.1 Hz) Precision Frequency Generator PTS-160 Programmed Test Sources, Littleton, MA, USA
PS1	0 - 30 amp dc, Current Regulated Power Supply 3002-1 Systron Donner, Alpha Subsidiary, Oakland, CA, USA
Magnet	800 A T^{-1} , 30 cm diameter x 45 cm long Solenoid Type Magnet Custom Component Fabricated by Joseph P. Hornak, Rochester Institute of Technology, Rochester, NY, USA
Lockin Amp.	Lockin Amplifier SR-530 Stanford Research Systems, Inc., Sunnyvale, Ca, USA

LFESR Component List, continued.	
Component	Description, Part Number, and Source
Computer	486 DX2-66 computer. Zeos International Limited, St. Paul, MN, USA
Digital I/O	(3) 8-Bit Digital I/O Ports of: Win-30 Omega Engineering Inc., Stanford, CT, USA
A/D Converter	0-5 v, 12-bit, Analog-to-Digital Converter of: Win-30 Omega Engineering Inc., Stanford, CT, USA
D/A Converter	0-5 v, 16-bit, Digital-to-Analog Converter of: Win-30 Omega Engineering Inc., Stanford, CT, USA
RS232 Port	Factory Installed RS232 Communications Port Zeos International Limited, St. Paul, MN, USA
10 kHz RF Source	DC to 8 MHz Frequency Synthesizer Multigen Frequency Synthesizer Sciteq Electronics, Inc., San Diego, CA, USA
Cable Configuration Converter	Hard-Wire, 50-pin Cable Configuration Converter (Figure 14) Custom Component Fabricated by: William J. Ryan Rochester Institute of Technology, Rochester, NY, USA
STS Sample Probe	248 MHz,15 ml, Single-Turn-Solenoid Sample Probe (Figure 15) Custom Component Fabricated by: William J. Ryan Rochester Institute of Technology, Rochester, NY, USA
Voltage Divider	Custom 0-5 v Volage Divider (Figure 16) Fabricated by: William J. Ryan Rochester Institute of Technology, Rochester, NY, USA

Appendix B User Instructions for the LFESR Spectrometer.

Contents **Page #**

Hardware Configuration

The LFESR spectrometer is a modular system composed of several integrated components. The arrangement and interconnection of the various components is illustrated in the following figures.

Figure B-1 illustrates the front view of the LFESR work station, with the respective locations of each of the major system components. References to these components will be made by the names listed in Figure B-1. For a list of the various components, with model numbers and sources, see Table A-1. Figures B-2 through B-9 illustrate the inter-connections that are made between the major system components. All RF connections are made with RG-58 coaxial cable, and BNC connectors.

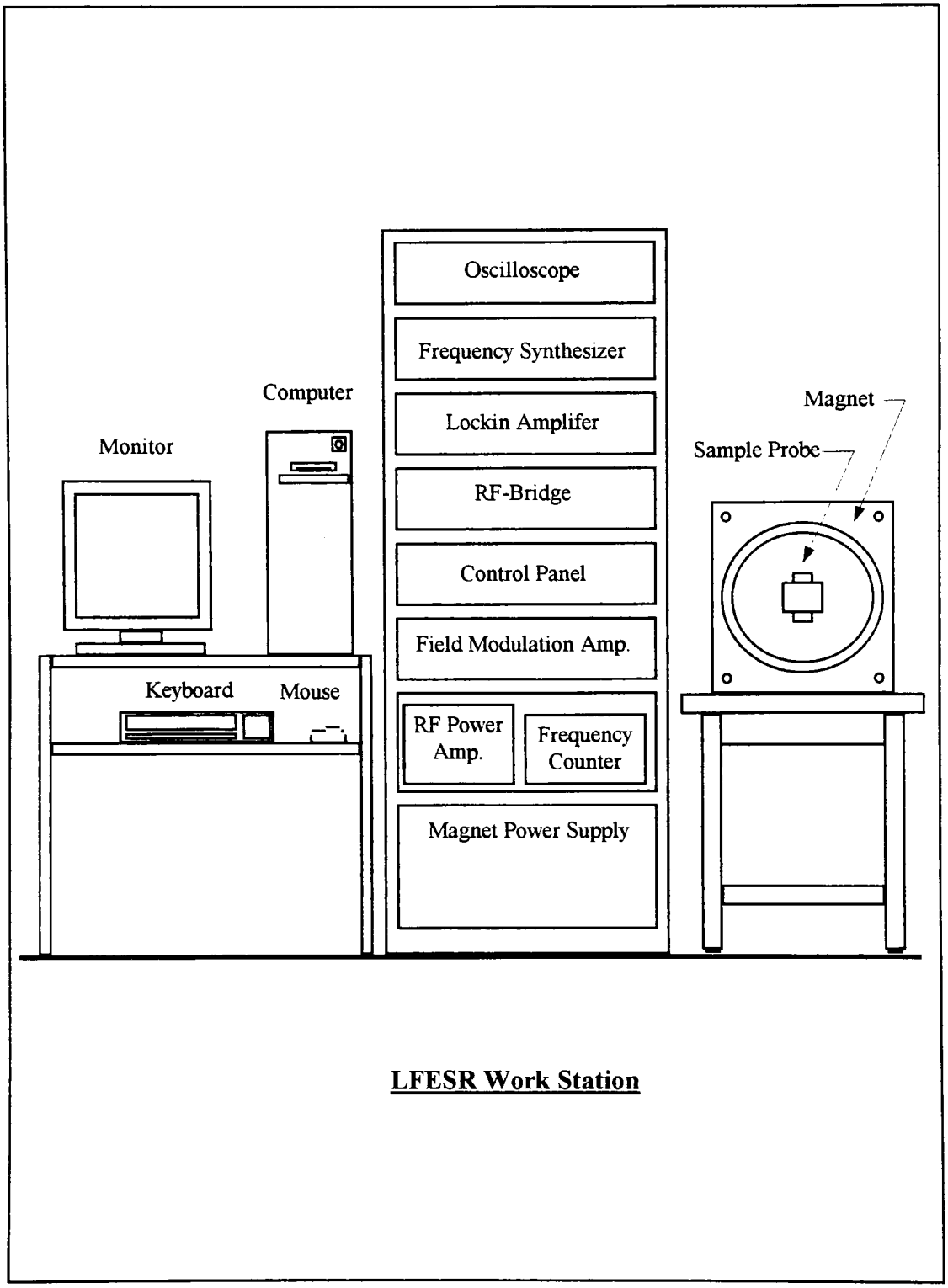


Figure B-1 Front view of the LFESR workstation. The major components are arranged as illustrated.

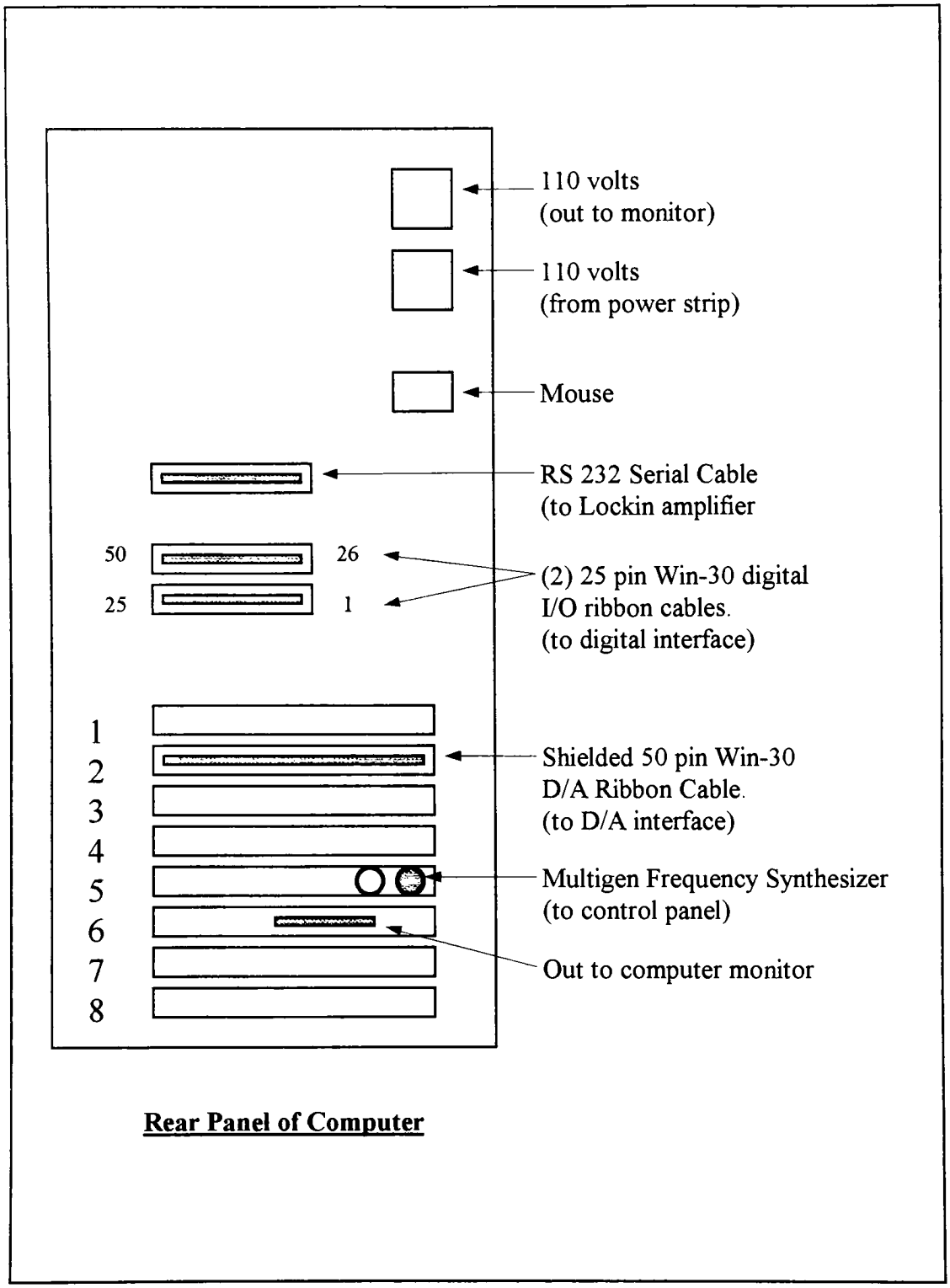
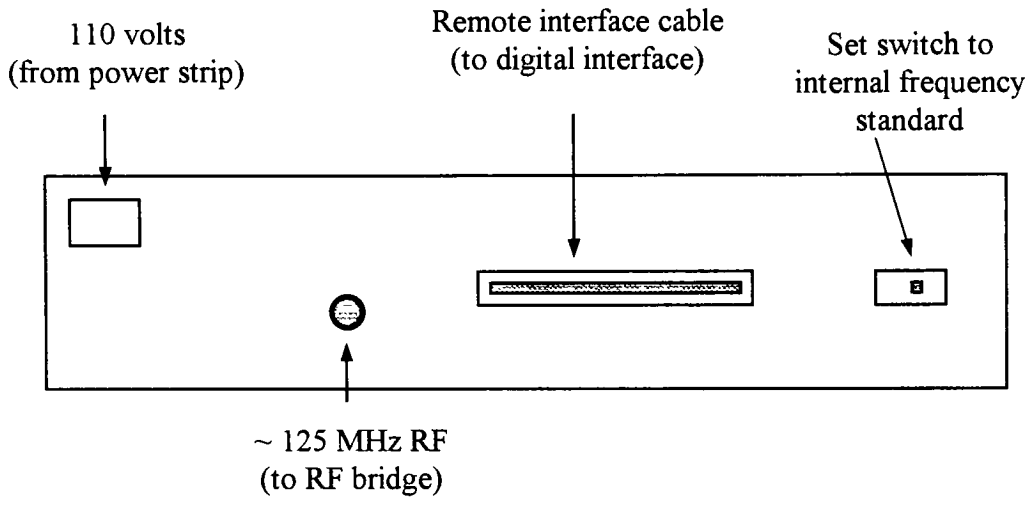


Figure B-2 Connections made at the computer of the LFESR spectrometer.



Rear Panel of the PTS-160 Frequency Synthesizer.

Figure B-3 Connections made at the PTS-160 frequency synthesizer of the LFESR spectrometer.

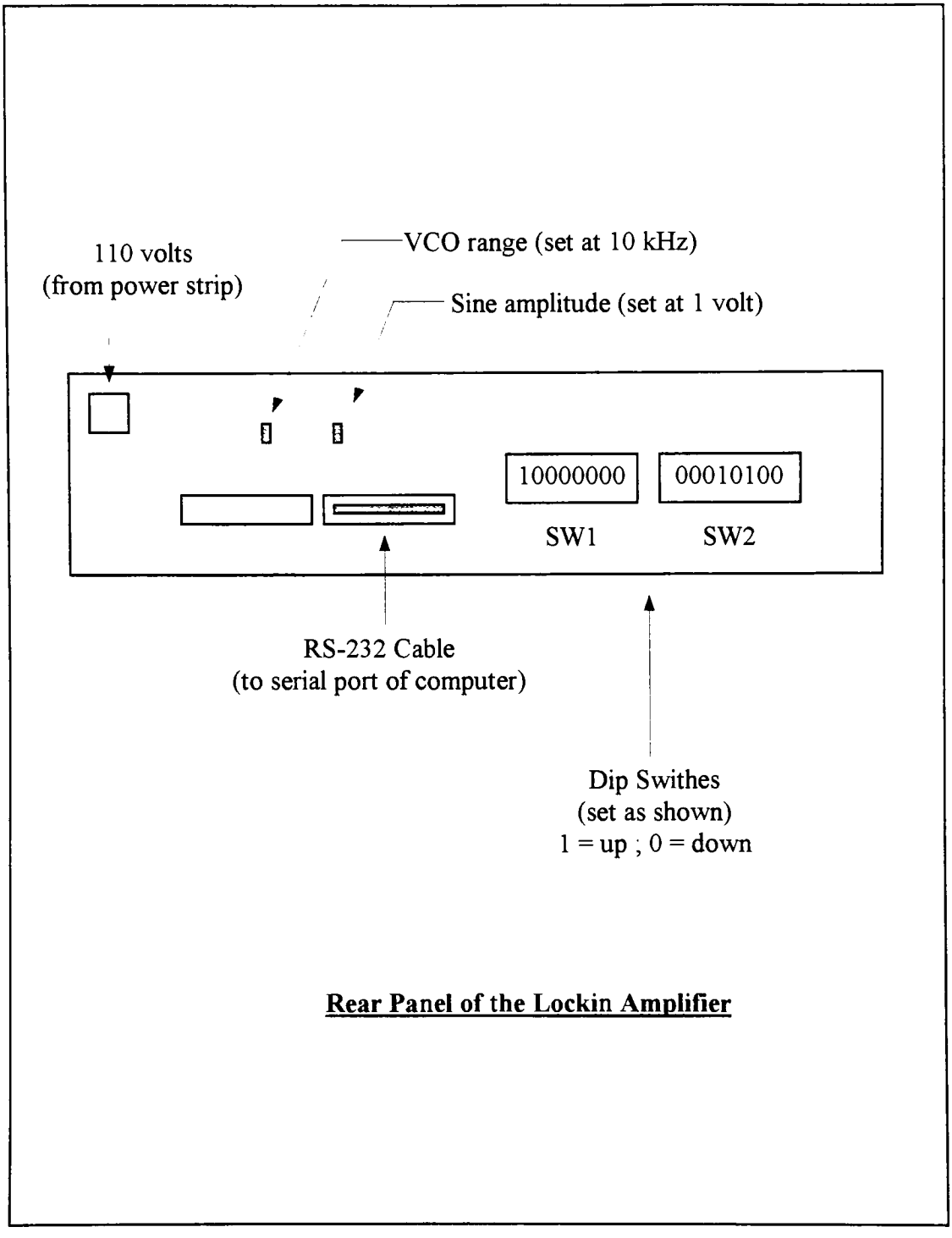


Figure B-4 Connections made at the SR-530 Lockin amplifier of the LFESR spectrometer. (Not shown: A 10 kHz RF reference is connected at the front face of the Lockin amplifier at the reference input. The reference is from the control panel.)

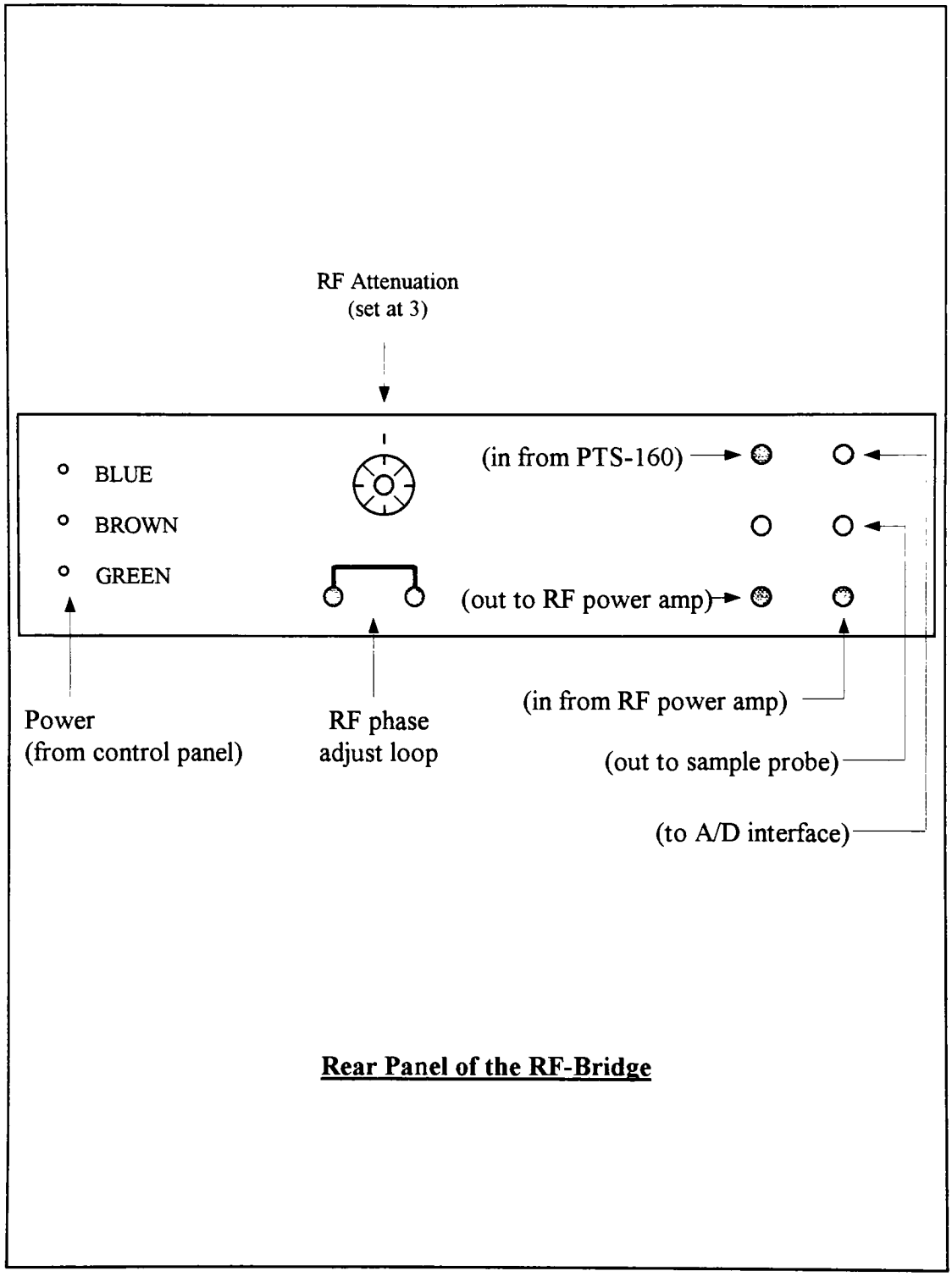


Figure B-5 Connections made at the RF-Bridge of the LFESR spectrometer.

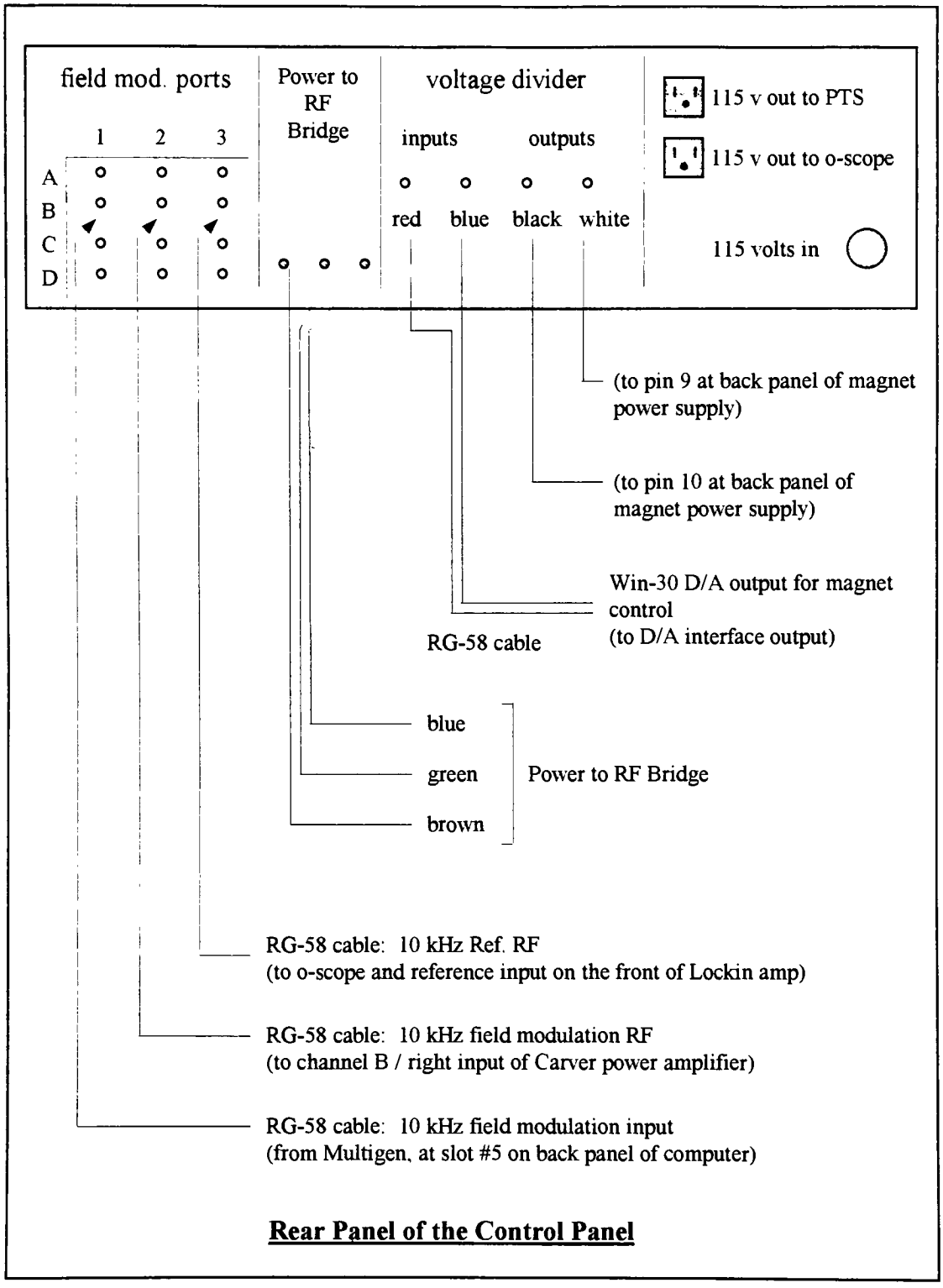
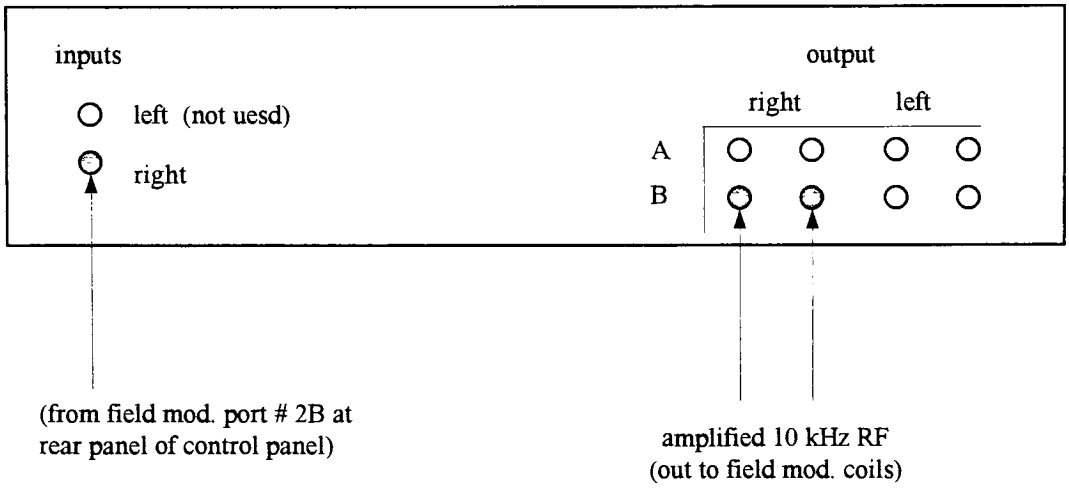
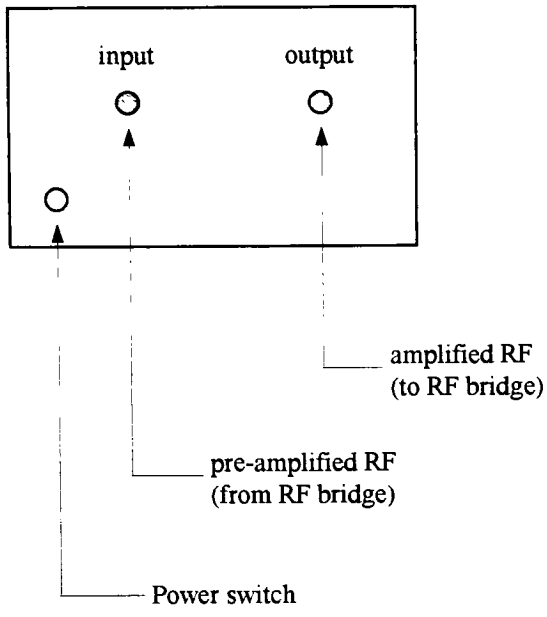


Figure B-6 Connections made at the Control Panel of the LFESR spectrometer.



Rear Panel of Field Modulation Amp.

Figure B-7 Connections made at the rear panel of the Field Modulation Amp.



Front Panel of RF Power Amp

Figure B-8 Connections made at the front face of the RF Power Amp.

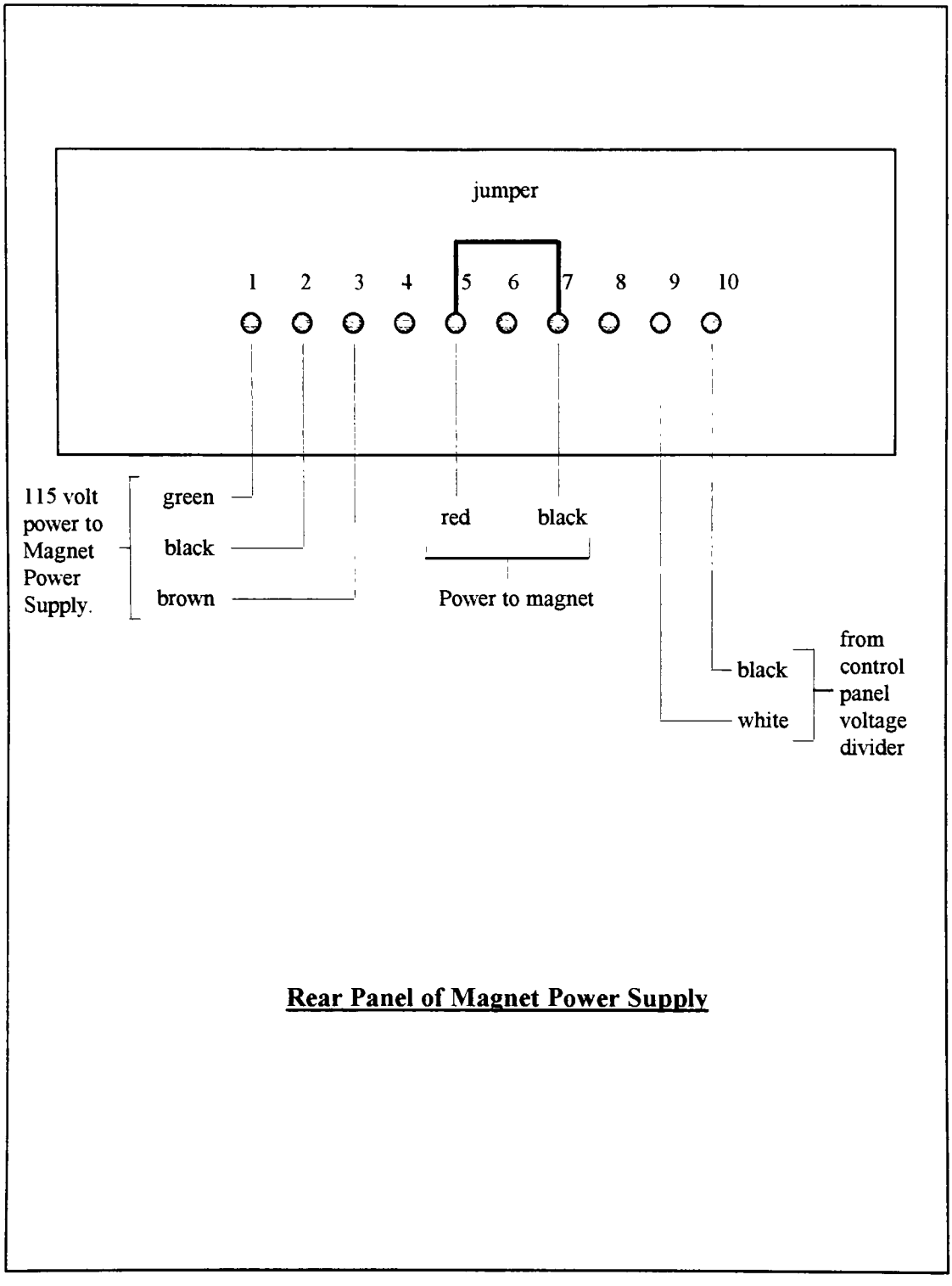


Figure B-9 Connections made at the rear panel of the Magnet Power Supply.

Software Configuration

Operation of the LFESR spectrometer requires that the LFESR Operating Software be loaded onto the hard-drive of the computer. The operating software is located on a single 3.5" disk file, and is loaded as described below.

- Start Window's and enter into **Program Manager**.
- Insert the LFESR Operating Software disk into the floppy disk drive.
- From the File menu of Program Manager, choose **Run** ...
- At the command line of the Run dialog box, type **b:setup**, and click on **OK**,
- You will be prompted to press any key to continue. **Press any key**. (If you wish to abort the setup, then press Ctrl Break.)
- When the files are loaded, you will be prompted to press any key. **Press any key**. The installation will be finished at this point.

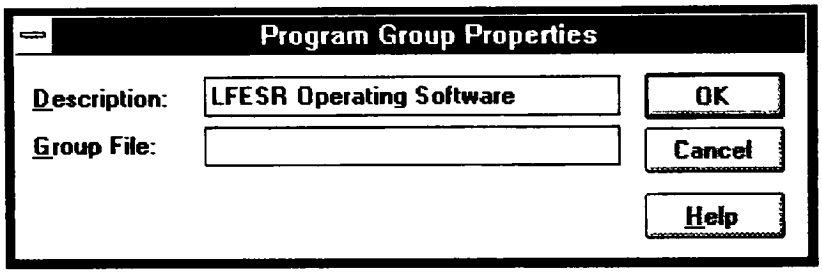
Building the LFESR Program Group Window with Start-up Icons.

After loading the LFESR Operating Software, build the LFESR program group, as follows.

- Start Window's (if it is not already started).

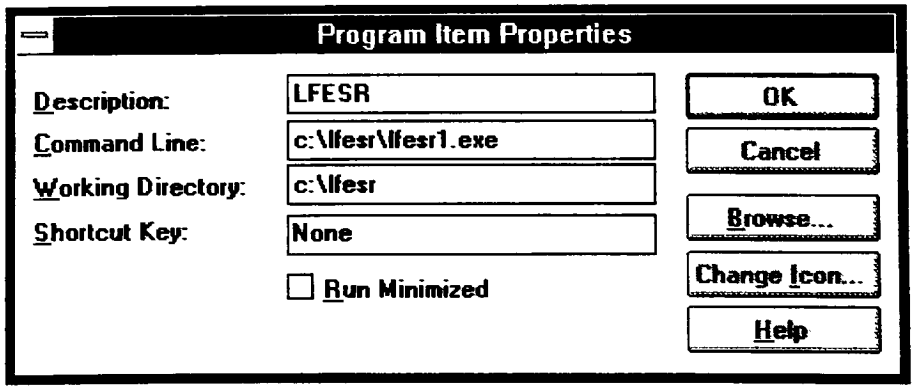
  Within **Program Manager** ...

  - From the **File** menu, choose **New...**
  - Select **Program Group**
  - Type the information into the **Program Group Properties** dialog box, as shown. Then click on **OK**.

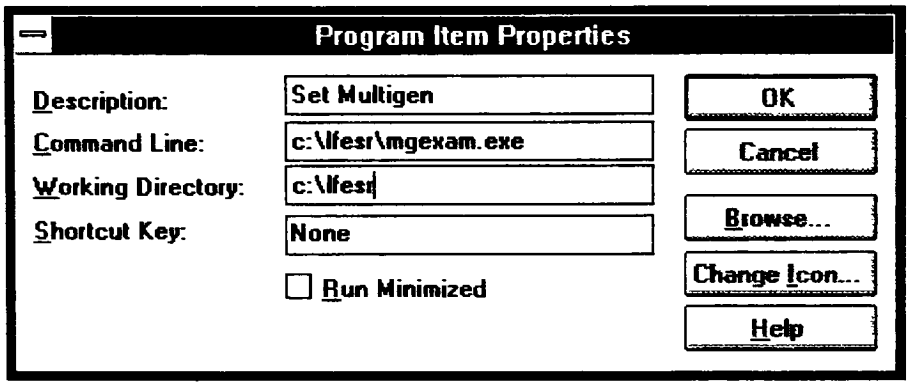


After building the **LFESR Program Group**, add start-up icons for the **LFESR1** and **Set Multigen** software , using the following procedure, for each icon added.

- Highlight the LFESR program group, and from the **File** menu, choose **New...**
- Select **Program Item** from the resulting dialog box, click on **OK**.
- Type the information into the **Program Item Properties** dialog box, as shown. Then click on **OK**.



- The **LFESR Program Group** should now contain an icon for the **LFESR1** software.
- Repeat the procedure to add an icon for the **Set Multigen** software, typing the following information into the Program Item Properties dialog box.

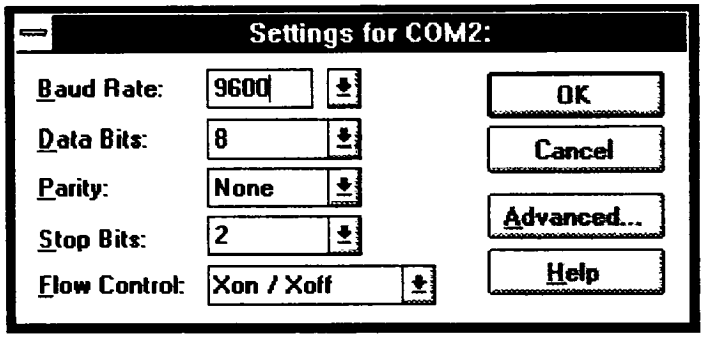


- At this point there should be an **LFESR** program group, containing start-up icons for the **Set Multigen** and **LFESR1** software.

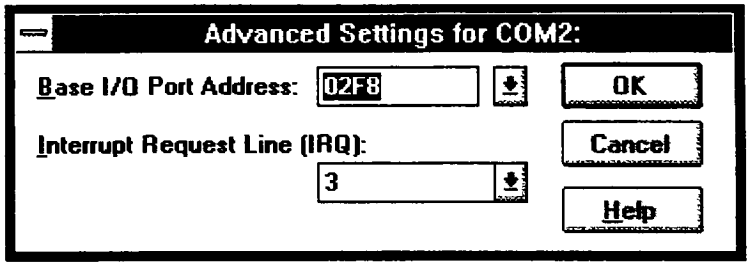
Computer RS232 Port Configuration

The software is written to utilize COM 2 for RS232 communications with the Lockin Amplifier. Before running the LFESR Operating Software, Comm 2 must be configured as follows.

- Start Window's.
- In the **Main** program group, double click on the **Control Panel** icon.
- In the resulting **Control Panel** group, double click on the **Ports** icon.
- In the resulting **Ports** dialog box, single click on **COM2**, and the single click on **Settings**.
- Each setting for **COM2** should be as shown. After the proper settings are made, single click on **Advanced...**



- The **Advanced Settings for COM2** dialog items should be set as shown. After the proper settings are made, click on **OK**.



- In the **Settings for COM2** dialog box, click on **OK**.
- In the **Ports** dialog box, click on **Close**.
- At this point, COM2 is ready for use by the LFESR Operating Software.

System Power-Up Procedure

Power up the system components in the following order. The power-up switches for each component are identified by a red dot. Eah of these switches should be in the position indicated by the red dot.

- Oscilloscope,
- PTS-160 Frequency Synthesizer,
- SR-530 Lockin Amplifier,
- RF Bridge (with RF attenuators set at 10 dBs, each),
- Control Panel (set switches at red dot positions, and potentiometer at 600Ω),
- Carver amp (A off, B on; Power out-put set to lowest position for A and B),
- ENI RF Power Amplifier,
- Magnet Power supply,
- Computer.

Starting the LFESR Operating Software

Wait for computer to boot-up, start windows, and then initialize the Multigen (10 kHz field modulation RF source) as follows.

- Double-click on the SET MULTIGEN icon, in the LFESR group.
- Enter 795 for the board address.
- Select "Frequency" (option 3), and enter a value of 10000.0 for the frequency (Note: The decimal point and zero must be included for the software to recognize the input.)
- Select "Level" (option 4), and enter a value of 0.5 for the level.
- Leave all other parameters alone (i.e. Leave at the default settings.)
- Exit the Multigen software by pressing "0".
- Verify the frequency and level settings of the Multigen, at the oscilloscope.

After initializing the Multigen as described above, proceed as follows, to initialize the LFESR operating software:

- Double-click on the LFESR icon in the LFESR group (This opens 3 nested windows).

  After the LFESR software starts, maximize the LFESR window, by clicking on the maximize button at the top-right corner of the window.
- Select "Tile" from the "Window" menu (This will tile the program windows, giving a graphics window on the left side of the screen, and a Stdin/Stdout/Stderr window on the right side of the screen).
- Click anywhere on the "Stdin/Stdout/Stderr" window to activate it.
- Press "1" followed by <ENTER>
- The return message should be "idCommDev = 1", indicating an opened comm

port, for serial communications with the Lockin amplifier.

- If a comm port is not found, then exit the software by entering "7", close the LFESR window, and start over by double clicking on the LFESR icon again, and repeating the above procedure (If one or two repeated attempts fail, then check to see if comm port 2 is properly configured for the software. See the section on "Computer RS232 Port Configuration" in this appendix).
- If the "Win-30 faulty or not found" message appears on the screen, exit the software by pressing "7", close the LFESR window, and start over by double clicking the LFESR icon and repeating the above procedure (If one or two repeated attempts fail, exit the software, close the LFESR window, exit Window's, and shut the system down as described in the "System Shutdown Procedure". Disconnect the power to the computer, remove the cover, and check to see if the Win-30 is properly installed.).

Tuning the Sample Probe

The sample probe is tuned as follows.

- Fill the sample vial with the sample solution. Care should be taken to keep the outside of the sample vial clean, to avoid contamination of the spectrometer. If the sample liberates gaseous products, the cap to the sample vial should be allowed to breathe. Place the sample vial in the sample probe, allowing the cap of the vial to rest on the top edge of the PVC form of the sample probe.
- Check to be sure that the RF attenuators on the front face of the RF-bridge are each set at 10 dB of attenuation, for a total of 20 dB.
  Select the "Tune Probe" option from the "main-menu" (option 2), and watch the graphics screen for the real-time frequency response of the probe to the swept range of frequencies. Repeat the probe-tuning sweep, while adjusting the inductive coupling of the probe by adjusting the plastic screw located on the top of the probe, next to, and parallel to the PVC form.
- The inductive coupling should be adjusted until the resulting probe-tuning sweep shows a maximum that touches the horizontal line at the top of the curve-trace plot. When this is accomplished, the probe is tuned and the system is ready for an ESR scan.

Note: The sweep parameters of "center frequency, Frequency width, step size, and number of sweeps" can each be varied as desired through the "change parameters ... probe-tuning" option. If no "peak" is observed in the probe-tuning sweep, the sweep parameters may be adjusted to more effectively bracket the resonant frequency of the probe. Also, adjusting the inductive coupling can give rise to an otherwise un-observed tuning peak.

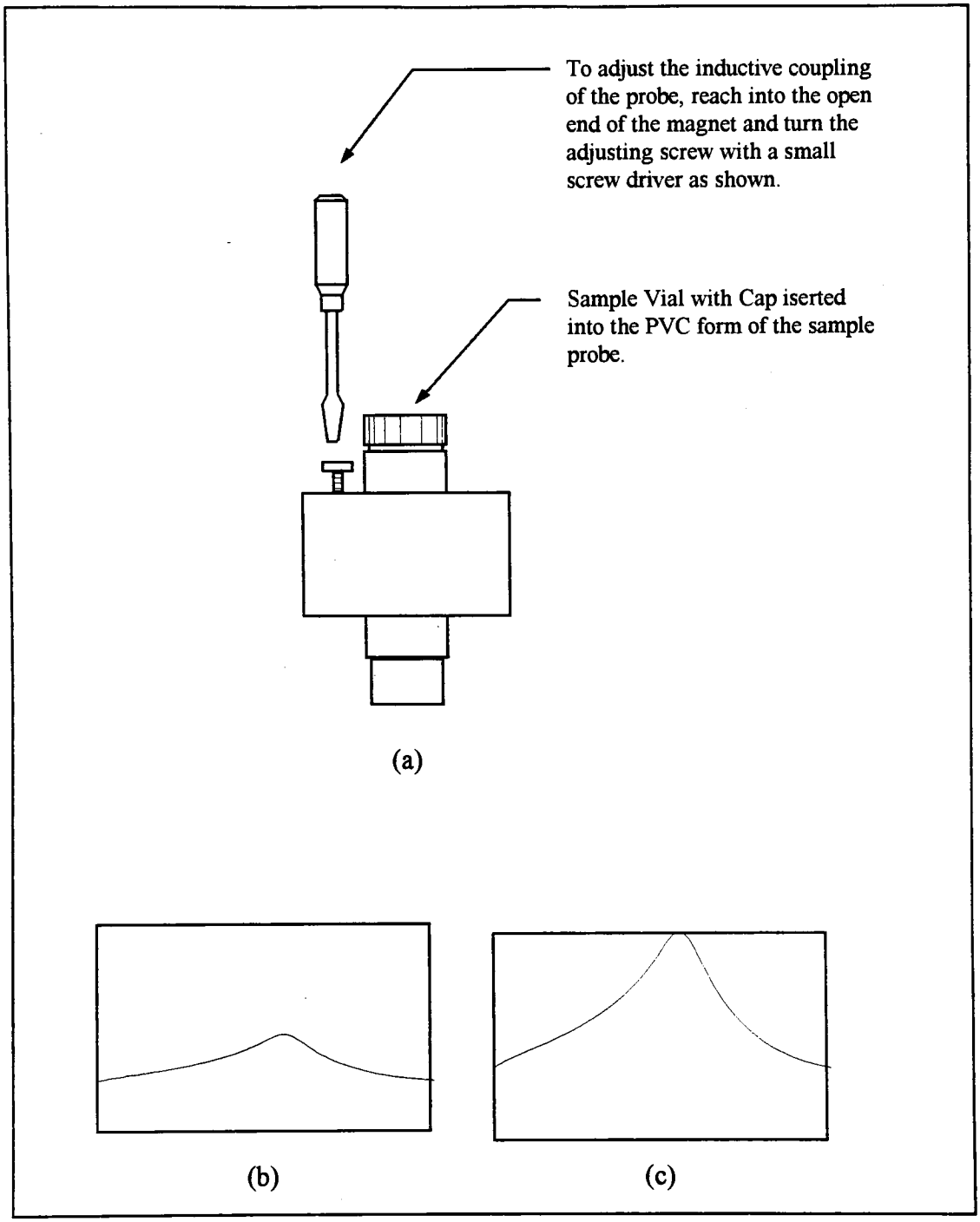


Figure B-10 Adjusting the inductive coupling of the sample probe. (a) Adjustments are made by turning the screw as shown, clockwise or clockwise. (b) A probe tuning sweep in which the probe is either over-coupled or under-coupled. (c) A probe tuning sweep in which the probe is optimumly coupled.

Recording a Spectrum

An ESR spectrum is recorded as follows. The spectrum is displayed in the "graphics" window of the LFESR operating software. The ESR data is stored in a data buffer. The data is retrieved and put to a disk file, if the user chooses the option to save the data after the spectrum has been acquired.

- Tune the probe as described.
- Set the desired sweep parameters by choosing the "New parameters ... Magnetic field" option. The sweep parameters are "Sweep Center, Sweep Width, and Step Size" If the parameters are not set by the user, a default set will be used by the program, to sweep from 84 to 92 Gauss, in steps of 0.1 Gauss.
- Set the output level of the Carver field modulation power amplifier (channel B) to 0.1 watts / 8 ohms. Do this by adjusting the knob for channel B, and watching the analog level meter, both located on the front face of the Carver field modulation power amplifier.
- Set the 0-70 dB and 0-10 dB RF power attenuators on the front face of the RF-Bridge to 0 and 10 dB, respectively.

  !!! The operator should NEVER adjust both attenuators to zero, simultaneously. This can result in damage to the instrumentation.
- Choose the "record spectrum" option from the main menu, and wait for the spectrum to be scanned. As it is scanned, the spectrum will be displayed in real time on the graphics window of the LFESR window.
- The resulting spectrum can be saved in the form of numeric data to a disk file on a drive of the user's choice, by answering "yes" for the software to save the spectrum. The numeric data is easily accessed for later processing inside of any Window's spreadsheet application.

Processing Data

ESR data, as well as probe-tuning data can be saved to a disk file and retrieved for processing in other Window's applications, as follows.

- Save the data to a disk file when prompted. The data is saved by typing the information as follows: drive:name.extension. The extension is optional.
- Retrieve the data from the disk file, through Window's "notebook" Select the file contents with "edit ... select all", and "edit ... copy"
- Paste the data into a spreadsheet of choice, and process it as desired.

Exiting the LFESR Operating Software

The LFESR operating software should be exited in the proper way, to avoid leaving the system components in an undesired rest state.

- From the main menu of the operating software, choose "exit".
- Close the LFESR window.

System Shut-Down Procedure

The components of the LFESR spectrometer are shut down in the following order. As with the system power-up procedure, the power switches are identified by red dots.

- Magnet Power supply
- ENI RF Power Amplifier
- Carver amp (A off, B on; Power out-put set to lowest position for A and B)
- Control Panel (set switches at red dot positions, and potentiometer at 600Ω)
- RF Bridge (with RF attenuators set at 10 dBs, each)
- SR-530 Lockin Amplifier
- PTS-160 Frequency Synthesizer
- Oscilloscope
- Computer

Appendix C -
Microsoft Visual C++ Program for the LFESR Spectrometer.

```
#include<windows.h>              // NOTE: Must be included above pc30.h
#include <time.h>
#include<iostream.h>
#include <conio.h>
#include<graph.h>
#include<stdlib.h>
#include<pc30.h>
#include<math.h>

#include<fstream.h>
#include<stdio.h>

#define  addatl          0x700
#define  addsr           0x701
#define  adccr           0x702
#define  admde           0x703
#define  gmem0           0x718
#define  gmem1           0x719
#define  gmem2           0x71A
#define  gmem3           0x71B
#define gainreg          0x718
#define diocntrl  0x70B

int main_choice;

float fdummy;
int idummy;
short sdummy;
long int lidummy;

int base = 0x700;
int diagnostics;

int da_channel = 0;
long number = 0;



long int frequency;
int dummy;
```

```
int num_of_tuning_sweeps = 1;
char num_of_tuning_sweeps_input [20];

long int freq_step_counter = 0;

long int center_freq = 248000;
long int freq_step = 20;
long int width = 4000;
short tuning_gain = 1;

char center_freq_input [20];
char freq_step_input [20];
char width_input [20];
char tuning_gain_input [20];

char character;

int idComDev;
        DCB dcb;

int err;

char lockin_data[16];
char lockin_string1[61] = "B1;L1,1;L2,1;D0;P+88;M0;E1,0;E2,0;T1,6;T2,0;G24;AX;AY;S0;I1\r";
char lockin_string2[61] = "B1;L1,1;L2,1;D0;P+88;M0;E1,0;E2,0;T1,6;T2,0;G10;AX;AY;S0;I1\r";

char output_string[4] = "Q1\r";



int samples_per_step = 1;
char samples_per_step_input [8];

float sweep_start_correction = 0.0;
float sweep_start = 84.0;
float sweep_width = 8.0;
float sweep_resolution = 0.10;
float steps_per_scan = sweep_width / sweep_resolution;
long signal_gain = 10000000;

char sweep_start_correction_input [20];
char sweep_start_input [20];
char sweep_width_input [20];
char sweep_resolution_input [20];
char signal_gain_input [20];

int choice = 0;


short probe_tune_data[500];
```

```
float b_min, b_max;
float derivative_amplitude, intensity;
double derivative_width;
float dY_min = 10000.0;
float dY_max = 0.0;
float relative_b = 0.0;
float esr_signal;
float signal[2000];

//*************************************************************************
//INITIALIZE  WIN-30

int diag_err;

initialize_win_30()
{

        set_base(base);
        diag_err = diag();          //performs diagnostics on WIN-30
        if (diag_err != 0)
        {
                cout<<"ERROR (WIN-30 faulty or not found)\n";
    return 0;
        }

        init();



                                        //initializes the board to a known state.
        d_mode(0,0,0);  //sets the 3 I/O ports to be outputs (mode=0)


        d_out(2,67);    //Remote enable ... a and b latches set low, c is high

        d_out(0,255);   //Sets the PTS to 124000 kHz (a safe value for the LFESR)
        d_out(1,191);
        d_out(2,115);   //                                  ... a and b latches set high, c is high

        d_out(2,67);    //          ... a and b latches set low, c is high

        return 0;
}


//*************************************************************************
// Initialize Lock-In Amlifier
```

```
//*************************************************************************

open_com_2()
{
        //****************************

        typedef struct tagDCB       /* dcb */
        {
          BYTE Id;
          UINT BaudRate;
          BYTE ByteSize;
          BYTE Parity;
          BYTE StopBits;
          UINT RlsTimeout;
          UINT CtsTimeout;
          UINT DsrTimeout;

          UINT fBinary      :1;
          UINT fRtsDisable  :1;
          UINT fParity      :1;
          UINT fOutxCtsFlow :1;
          UINT fOutxDsrFlow :1;
          UINT fDummy       :2;
          UINT fDtrDisable  :1;

          UINT fOutX        :1;
          UINT fInX         :1;
          UINT fPeChar      :1;
          UINT fNull        :1;
          UINT fChEvt       :1;

          UINT fDtrflow     :1;
          UINT fRtsflow     :1;
          UINT fDummy2      :1;

          char XonChar;
          char XoffChar;
          UINT XonLim;
          UINT XoffLim;
          char PeChar;
          char EofChar;
          char EvtChar;
          UINT TxDelay;
        } DCB;


        //****************************


        DCB dcb;
```

```
        int err;

        //*****************************

        idComDev = OpenComm("COM2", 1024, 128);
        if (idComDev < 0)
        {
        cout<<"ERROR (OpenComm) = ("<<err<<")\n";
        return 0;
        }
        else
        {
                cout<<"idComDev = "<<idComDev<<"\n";
        }

        //*****************************

        err = BuildCommDCB("COM2:9600,n,8,2", &dcb);
        if (err < 0)
        {
        cout<<"ERROR (BuildCommDCB) = ("<<err<<")\n";
        return 0;
        }

        //*****************************

        err = SetCommState(&dcb);
        if (err < 0)
        {
                cout<<"ERROR (BuildCommDCB) = ("<<err<<")\n";
    return 0;
        }


  return 0;
}
//*****************************************************************************


close_com_2()
{
  CloseComm(idComDev);

        return 0;
}

//*****************************************************************************

void delay(int delay_value)
```

```
{

        clock_t goal_time;
        goal_time = delay_value + clock();

        while( goal_time > clock() );        //sit here until the goal time is
                                                                    //greater than the
clock time.
}

//*************************************************************************



void    main_menu(void)
{
        cout << "M A I N   M E N U \n";

        cout << " \n";

        cout << "1        New Parameters ... \n";
        cout << "2        Tune Probe \n";
        cout << "3        Record Spectrum \n";
        cout << "4        \n";
        cout << "5        \n";
        cout << "6        List Current Parameters \n";
        cout << "7        Exit Program \n";

        cout << " \n";

        cout << "PLEASE  MAKE  YOUR  SELECTION \n";

        main_choice = 0;

        return;
}


//*************************************************************************

void    change_parameters_menu(void)
{

        cout << "C H A N G E   P A R A M E T E R S   M E N U \n";

        cout << " \n\n\n";

        cout << "1        Magnetic Field ...  \n";
        cout << "2        Probe Tuning ...  \n";
        cout << "3        Signal Averaging ...  \n";
```

```
        cout << "4       Sweep Start Correction ... \n";
        cout << "5       ESR Display Gain ... \n";
        cout << "6        \n";
        cout << "7        \n";
        cout << "8        \n";
        cout << "9       Return to Main Menu \n";

        cout << " \n";

        cout << "PLEASE MAKE YOUR SELECTION \n";

        return;

}

//*************************************************************************

void    yes_no_menu(void)
{


        cout << "1       YES \n";
        cout << "2       NO \n";
        cout << " \n";
        cout << " \n";

        cout << "PLEASE MAKE YOUR SELECTION \n";

        return;

}


//*************************************************************************
//This module Clears the A/D sub-system.
//See Win-30 user manual, p96
//*************************************************************************

clear_ad_subsystem()

{



        _outp(admde,0x92);

        _outp(adccr,0x2);
```

```
        _inpw(addatl);

        return 0;
}


//*************************************************************************
//This function sets the gain on all A/D channels to zero
//See Win-30 user manual  p98
//*************************************************************************

set_gain()

{
        outp(gmem0,0);
        outp(gmem1,0);
        outp(gmem2,0);
        outp(gmem3,0);

        return 0;
}

//*************************************************************************
//This sets the channel to be converted,and
//sets the "strobe-select" bit of the ADCCR.
//*************************************************************************

set_channel_and_strobe_select()

{
        _outp(adccr,2);   //set the channel to be ch0, and set the strobe select

        return 0;
}

//*************************************************************************
//Toggel the software strobe bit to result in a single A/D conversion
//*************************************************************************

toggel_the_software_strobe_bit()

{
        _outp(adccr,3);   //toggel the software strobe bit to "high"
        _outp(adccr,2);   //toggel the software strobe bit to "low"

        return 0;
}
```

```
//*********************************************************************
//This function waits for the done bit to be set to 1.
//*********************************************************************

wait_for_done_bit()
{
        float done_bit;

        done_bit=_inp(admde)&64;

        do
        {
                done_bit=_inp(admde)&64;
        }
        while (done_bit!=64);


        return 0;
}


//*********************************************************************
//This reads the low byte and high nibble of the A/D data registers
//*********************************************************************

short read_ad_data()
{
        short y;
        float     word;

        word = (_inpw(addatl)&4095);
        word = (word-2048)*5/2048;
        //cout<<frequency<<"\t"<<word;


        word = 200.0f * word;
        y = (short)word;
        //cout<<"\t"<<y<<"\n";

        return y;
}


//*********************************************************************
//This sets the PTS-160 to a safe value for the ESR instrumentation.
//It also sets the PTS remote pin (#42) high, to enable remote control.
//It also sets the latches for the digits assiciated with the Win-30
//digital I/O ports A and B to high.  This keeps the data comming from
//ports A and B from being accepted by the PTS until port C data gets
```

```
//sent to the PTS (with the latches set low).
//*************************************************************************
void set_pts_initial()
{
        _outp(0x708,255);
        _outp(0x709,191);
        _outp(0x70A,115);               //latches low

        _outp(0x708,255);
        _outp(0x709,191);
        _outp(0x70A,67);                //latches high

        return;
}

//*************************************************************************
//This sets the output frequency of the PTS-160.
//*************************************************************************

void set_pts(long int frequency)

{
        long int freq;
        long int _10MHzDigit;
        long int _1MHzDigit;
        long int _100KHzDigit;
        long int _10KHzDigit;
        long int _1KHzDigit;

        long int A_low;
        long int A_high;
        long int B_low;
        long int B_high;
        long int C_low;
        long int C_high;

        long int A;
        long int B;
        long int C;



        freq = frequency;


        _10MHzDigit = freq/10000;                                   //This breaks the
frequency into
        _1MHzDigit = (freq/1000)-(freq/10000*10);   //separate digits.
        _100KHzDigit = (freq/100)-(freq/1000*10);
```

```
        _10KHzDigit = (freq/10)-(freq/100*10);
        _1KHzDigit = (freq)-(freq/10*10);



        A_low  = _1KHzDigit;    //This renames variables for the sake of clarity in
        A_high = _10KHzDigit;   //subsequent code.
        B_low  = _100KHzDigit;
        B_high = _1MHzDigit;
        C_low  = _10MHzDigit;
        C_high = 8;                                  //This sets the latches low, allowing new data
                                                            //to be accepted by the PTS.



        A = 255 - (A_low + 16*A_high);  //This combines the 4-bit chunks of data
        B = 255 - (B_low + 16*B_high);  //into 8-bit bytes, and also gets the data
        C = 255 - (C_low + 16*C_high);  //into a negative-true format for the PTS.


  //cin>>dummy;
        _outp(0x708,(int)A);        //This will send the data to the PTS.

        //cin>>dummy;
        _outp(0x709,(int)B);

        //cin>>dummy;
        _outp(0x70A,(int)C);

        //cin>>dummy;
        C_high = 11;      //This sets the latches high for ports A and B.
        C = 255 - (C_low + 16*C_high);
        _outp(0x70A,(int)C);




return ;
}


//*************************************************************************

void tuning_plot(short x,short y, short yy)
{
        float tune_x_step = (float)200 / (float)(width/freq_step);

        x=(short)(freq_step_counter * tune_x_step + 20);
        y=y+20;
        yy=yy+20;
```

```
        //_rectangle(_GBORDER, x-1,y-1,x+1,y+1);
        //_rectangle(_GFILLINTERIOR, x-1,yy-1,x+1,yy+1);

        _setpixel( x,y);
        _setpixel( x,yy);

        return;
}


//*********************************************************************
//TUNE PROBE
//*********************************************************************


long int sweep()


{
        long int resonance_freq;
        short voltage_minimum = 9999;
        short x,y,yy;
        set_pts_initial();


        //Calculate the scan-loop parameters


        long int low_freq, high_freq, temp_x;

        //*********************************************************
        // This divides the sweep input parameters before setting the PTS,
        // since the output of the PTS goes to a frequency doubler.
                center_freq = center_freq / 2;
                width = width / 2;
                freq_step = freq_step / 2;

                low_freq = center_freq - (width/2);

                high_freq = center_freq + (width/2);
        //*********************************************


        //Initialize the A/D subsystem.


        clear_ad_subsystem();

        set_gain();
```

```
        set_channel_and_strobe_select();


        /* Find a valid graphics mode. */
        if( !_setvideomode( _MAXRESMODE ) )
exit( 1 );
        _rectangle( _GBORDER, 20, 20, 220, 220 );

      //*********************************************
                frequency = 1;

                                set_pts(frequency);

                                delay(50);

                                toggel_the_software_strobe_bit();

                                wait_for_done_bit();

                                y = read_ad_data();

                                yy = y;              // yy = the base-line voltage.
             //*********************************************

                frequency = low_freq;
                freq_step_counter = 0;

                do
                {
                        set_pts(frequency);

                        delay(60);

                        toggel_the_software_strobe_bit();

                        wait_for_done_bit();

                        y = read_ad_data()*tuning_gain;

                        temp_x = (frequency - low_freq)/2;

                        x = (short)temp_x;

                        tuning_plot((short)freq_step_counter,y,yy);


                        probe_tune_data[freq_step_counter] = y-yy;
```

```
                                        if (y < voltage_minimum)
                                        {
                                        voltage_minimum = y;
                                        resonance_freq = frequency;
                                        }


                        frequency = frequency + freq_step;
                        freq_step_counter = freq_step_counter+1;




                }
                while (frequency <= high_freq);


        frequency = resonance_freq;

        //*********************************************************
        // This restores the sweep parameters, after using them to set the PTS.
        // which allows the parameter output values to reflect the un-divided
        // values.

                center_freq = center_freq * 2;
                width = width * 2;
                freq_step = freq_step * 2;

        //*********************************************

        return frequency;
}

tune_probe()
{

        /* Find a valid graphics mode. */
        if( !_setvideomode( _MAXRESMODE ) )
   exit( 1 );
        _rectangle( _GBORDER, 20, 20, 220, 220);



        int counter = 1;
        do
        {
```

```
frequency = sweep();

counter ++;
}
while (counter <= num_of_tuning_sweeps);

cout<<"Resonance Frequency of Probe = "<<2*frequency <<"\n";

set_pts(frequency);

//*********************************************************************
//This section allows the user to save the probe tuning data to a disk file

        ofstream fstr;
int     choice = 0;
char choice_input = 0;


do
{
        cout<<"Save the probe-tuning sweep data ? \n";
        yes_no_menu();
        cin >> choice_input;
        choice = (int) choice_input;
}
while (((int)choice < 49) || ((int)choice > 50));



switch(choice)
{



        case    49:
                //save data code
                char file_name[12];
                cout << "Enter a drive, filename, and extension\n";
                cin >> file_name;

                fstr.open(file_name, ios::out);
                if(!fstr.good())
                        cout << "ERROR opening file.\n";
                        else
                        {

                                fstr << "Center Frequency (kHz) = " <<
center_freq << "\n";
                                fstr << "Sweep Width (kHz)      = " << width
<< "\n";
```

```
                                                fstr << "Step Size (kHz)       = " << freq_step
<< "\n";
                                                fstr << "Resonant Freq (kHz)    = " <<
frequency << "\n";

                                                int ctr = 0;
                                                for(ctr = 0; ctr <= freq_step_counter-1; ctr++)
                                                {
                                                        fstr << probe_tune_data[ctr] <<"\n";
                                                }
                                        }

                                break;



                        case    50:
                                //
                                break;

                }

        return 0;
}
//*************************************************************************


//*************************************************************************

record_esr_spectrum()
{

   open_com_2();
        WriteComm(idComDev, lockin_string2, 60);

        delay(500);

//steps_per_scan = sweep_width / sweep_resolution;


float bfield_start = (sweep_start + (float)169.8 + sweep_start_correction) * ((float)200);

float bfield_stop = bfield_start + (sweep_width * (float)200);

float bfield_step = sweep_resolution * (float)200;
```

```
float x_step = (float)256 * bfield_step / (bfield_stop - bfield_start);


        /* Find a valid graphics mode. */
        if( !_setvideomode( _MAXRESMODE ) )
    exit( 1 );
        _rectangle( _GBORDER, 5, 5, 261, 180 );
        _rectangle( _GBORDER, 5, 180, 261, 355 );
        _rectangle( _GBORDER, 133, 5, 133, 355 );




    //float data[256];

    short esr_x, esr_y;



        long da_code = (long)bfield_start;
        int esr_step_counter = 0;

                //*****************************************************
                //Let the magnet equilibrate at the initial value ...
                        da_out(0,(int)da_code);
                        delay (5000);
                //*****************************************************

        do
        {

        da_out(0,(int)da_code);

                int sample_counter = 1;
                float signal_sum = 0;

                        do
                        {
                                WriteComm(idComDev, output_string, 3);

                                        delay(250);

                                ReadComm(idComDev, lockin_data, 16);

                                signal_sum = signal_sum + float (atof(lockin_data));
```

```
                sample_counter = sample_counter + 1;
        }
        while (sample_counter <= samples_per_step);


esr_signal =  signal_sum / (float)samples_per_step;

signal[esr_step_counter] = esr_signal;

//*********************************************************
//Relative Intensity Determination

if (esr_signal < dY_min)
{
        dY_min = esr_signal;
        b_min = relative_b;
}

if (esr_signal > dY_max)
{
        dY_max = esr_signal;
        b_max = relative_b;
}

derivative_amplitude = dY_max - dY_min;
derivative_width = sqrt((double)(b_min - b_max)*(double)(b_min - b_max));

intensity = (dY_max - dY_min) / 2 * (b_min - b_max) * (b_min - b_max);

relative_b = relative_b + sweep_resolution;

//*********************************************************



esr_x = short(esr_step_counter * x_step + 5);
esr_y = short(180 - signal_gain * esr_signal);



_rectangle( _GBORDER, esr_x - 1, esr_y - 1, esr_x +1, esr_y +1);
//_setpixel( esr_x, esr_y);

esr_step_counter ++;

da_code = da_code + (long)bfield_step;
```

```
}
while (da_code <= bfield_stop);

cout << "DONE \n";

//*************************************************************
do
{
        da_out(0,(int)da_code);
        da_code = da_code - 100;
        delay(10);
}
while (da_code >= 32767);

da_out(0,32767);
//*************************************************************


WriteComm(idComDev, lockin_string1, 60);

                delay(500);

                //WriteComm(idComDev, output_string, 3);

                        //delay(250);

                //ReadComm(idComDev, lockin_data, 16);

                delay(250);

close_com_2();

//cout << "Intensity = " << intensity <<"\n";
cout << "\n";
cout << "Derivative Amplitude (v) = " << derivative_amplitude <<"\n";
cout << "Derivative Width (G)     = " << derivative_width;
cout << "\n\n";



        //****************
        //This code allows the user to save the esr data to a disk file,
        //under a user-designated file name, and drive.
                //*****************


                ofstream fstr;
                int     choice = 0;
```

```
char choice_input = 0;


do
{
        cout<<"Save the spectrum ? \n";
        yes_no_menu();
        cin >> choice_input;
        choice = (int) choice_input;
}
while (((int)choice < 49) || ((int)choice > 50));



switch(choice)
{


        case    49:
                //save data code
                char file_name[12];
                cout << "Enter a drive, filename, and extension\n";
                cin >> file_name;

                fstr.open(file_name, ios::out);
                if(!fstr.good())
                        cout << "ERROR opening file.\n";
                        else
                        {

                                fstr << "Sweep Start (G) = " << sweep_start
<< "\n";
                                fstr << "Sweep Width (G) = " << sweep_width
<< "\n";
                                fstr << "Sweep Resolution (G) = " <<
sweep_resolution << "\n";
                                fstr << "\n";
                                fstr << "Frequency (kHz) = " << 2*frequency
<< "\n";
                                fstr << "\n";
                                fstr << "Samples per Step = " <<
samples_per_step << "\n";
                                fstr << "Derivative Width (G) = " << intensity
<< "\n";
                                fstr << "Derivative Amplitude (v) = "<<
derivative_amplitude << "\n";


                                int ctr = 0;
```

```
                                        for(ctr = 0; ctr <= esr_step_counter-1; ctr++)
                                        {
                                                fstr << signal[ctr] <<"\n";
                                        }
                                }


                        break;




                case    50:
                        //
                        break;

        }


    //**********************************
    //Reset the intensity variables ...

                b_min, b_max;
                intensity = 0.0;
                dY_min = 10000.0;
                dY_max = 0.0;
                relative_b = 0.0;
        //**********************************

        return 0;
}

//*****************************************************************************
//*****************************************************************************
//
// M A I N
//
//*****************************************************************************
//*****************************************************************************

main()
{

        /* Find a valid graphics mode. */
        if( !_setvideomode( _MAXRESMODE ) )
    exit( 1 );
        _rectangle( _GBORDER, 20, 20, 220, 220);

                cout<<"SIZE AND TILE YOUR WINDOWS NOW ...\n";
```

```
                cin>>dummy;
                cout<<"\n";


        open_com_2();
        WriteComm(idComDev, lockin_string1, 60);


                        //delay(500);

                        //WriteComm(idComDev, output_string, 3);

                        //delay(500);

                //ReadComm(idComDev, lockin_data, 16);


  delay(500);
        close_com_2();

        initialize_win_30();



begining:

int     main_choice = 0;
int parameter_choice = 0;

char parameter_choice_input[30];
char main_choice_input[30];



        do
        {
                main_menu();

    fflush(stdin);
                cin >> main_choice_input;

                main_choice = atoi(main_choice_input);


        }
        while ((main_choice < 1) || (main_choice > 7));
```

```
switch(main_choice)
{
        case    1:

          do
          {
                change_parameters_menu();

                fflush(stdin);
                cin >> parameter_choice_input;
                        parameter_choice = atoi(parameter_choice_input);


                }
                while ((parameter_choice < 1) || (parameter_choice > 9));

                switch(parameter_choice)
                {

                case    1:

                        do
                        {
                                cout<<"Sweep Start (G) ... "<<sweep_start<<"\n";
                                cout<<"                ... ";
                                cin>>sweep_start_input;
                                fdummy = (float)(atof(sweep_start_input));
                                cout<<"\n";
                        }
                        while(fdummy<=0);
                        sweep_start = (float)(atof(sweep_start_input));

                        do
                        {
                                cout<<"Sweep Width (G) ... "<<sweep_width<<"\n";
                                cout<<"                ... ";
                                cin>>sweep_width_input;
                                fdummy = (float)(atof(sweep_width_input));
                                cout<<"\n";
                        }
                        while(fdummy<=0);
                        sweep_width = (float)(atof(sweep_width_input));

                        do
                        {
                                cout<<"Sweep Resolution (G) ...
"<<sweep_resolution<<"\n";
                                cout<<"                    ... ";
                                cin>>sweep_resolution_input;
```

```
                    fdummy = (float)(atof(sweep_resolution_input));
                    cout<<"\n";
                }
                while(fdummy<=0);
                sweep_resolution = (float)(atof(sweep_resolution_input));


                break;


        case    2:

                do
                {

                    cout<<"Center Frequency (kHz) ... " << center_freq
<<"\n";

                    cout<<"                    ... ";
                    cin>>center_freq_input;
                    lidummy = (long int)(atof(center_freq_input));
                    cout<<"\n";
                }
                while(lidummy<=0);
                center_freq = (long int)(atof(center_freq_input));




                do
                {

                    cout<<"Frequency Width (kHz) ... " << width <<"\n";
                    cout<<"                    ... ";
                    cin>>width_input;
                    lidummy = (long int)(atof(width_input));
                    cout<<"\n";
                }
                while(lidummy<=0);
                width = (long int)(atof(width_input));




                do
                {

                    cout<<"Frequency step (kHz) ... " << freq_step <<"\n";
                    cout<<"                    ... ";
                    cin>>freq_step_input;
                    lidummy = (long int)(atof(freq_step_input));
```

```
                    cout<<"\n";
                }
                while(lidummy<=0);
                freq_step = (long int)(atof(freq_step_input));



                do
                {

                    cout<<"Number of sweeps ... " <<
num_of_tuning_sweeps <<"\n";
                    cout<<"              ... ";
                    cin>>num_of_tuning_sweeps_input;
                    idummy = (int)(atof(num_of_tuning_sweeps_input));
                    cout<<"\n";
                }
                while(idummy<=0);
                num_of_tuning_sweeps =
(int)(atof(num_of_tuning_sweeps_input));


                do
                {

                    cout<<"Tuning gain ... " << tuning_gain <<"\n";
                    cout<<"          ... ";
                    cin>>tuning_gain_input;
                    sdummy = (short)(atof(tuning_gain_input));
                    cout<<"\n";
                }
                while(sdummy<=0);
                tuning_gain = (short)(atof(tuning_gain_input));


                break;


        case    3:

          do
                {

                    cout<<"Samples per step ... " << samples_per_step
<<"\n";
                    cout<<"                  ... ";
                    cin>>samples_per_step_input;
                    idummy = (int)(atoi(samples_per_step_input));
                    cout<<"\n";
                }
```

```
                    while(idummy<=0);
                    samples_per_step = (int)(atoi(samples_per_step_input));


                    break;


            case    4:

                    do
                    {

                            cout<<"Sweep start correction (G)... " <<
sweep_start_correction <<"\n";
                            cout<<"                          ... ";
                            cin>>sweep_start_correction_input;
                            fdummy = (float)(atof(sweep_start_correction_input));
                            cout<<"\n";
                    }
                    while(fdummy<=0);
                    sweep_start_correction =
(float)(atof(sweep_start_correction_input));


                    break;


            case    5:

                    do
                    {

                            cout<<"ESR Signal Gain ... " << signal_gain <<"\n";
                            cout<<"                    ... ";
                            cin>>signal_gain_input;
                            lidummy = (long)(atol(signal_gain_input));
                            cout<<"\n";
                    }
                    while(lidummy<=0);
                    signal_gain = (long)(atol(signal_gain_input));

                    break;



            case    6:
                    //
                    break;
                    //
```

```
            case    7:

                    //
                    break;

            case    8:
                    //


                    break;


            case    9:
              //
                    break;


}
            break;



case        2:
            tune_probe();

            break;


case        3:
            record_esr_spectrum();

            break;


case        4:

            //

            break;


case        5:
            //save_spectrum();
            break;
```

```
                case    6:
                        cout << "Magnetic Field \n";

                        cout << "   Sweep Start (G)....."<<sweep_start<<"\n";
                        cout << "   Sweep Width (G)....."<<sweep_width<<"\n";
                        cout << "   Sweep Resolution (G)...."<<sweep_resolution<<"\n";
                        cout << "   Samples Per Step....."<<samples_per_step<<"\n";
                        cout << "   ESR Display Gain....."<<signal_gain<<"\n";
                        cout << "   Sweep Start Correction.."<<sweep_start_correction<<"\n";
                        cout << "\n";

                        cout << "Probe Tuning \n";
                        cout << "   Center frequency (kHz).."<<center_freq<<"\n";
                        cout << "   Frequency step (kHz)...."<<freq_step<<"\n";
                        cout << "   Number of sweeps....."<<num_of_tuning_sweeps<<"\n";
                        cout << "   Tuning Gain....."<<tuning_gain<<"\n";

                        break;



                case    7:
                        goto end_of_program;
                        break;

        }


goto begining;

end_of_program:

open_com_2();

                WriteComm(idComDev, output_string, 3);

                        delay(250);

                ReadComm(idComDev, lockin_data, 16);

                delay(250);


cout<<"DONE .....";

close_com_2();

return 0;
```

Appendix D - Energy Diagrams for ·H and Fremy's Salt.

The energy levels for ·H and Fremy's salt are described by Equations D-1 through D-4, and D-5 through D-10, respectively. The derivations of these equations are available to the interested reader (Poole, 1987). These equations are used to generate energy level diagrams (Figures D-1 and D-3), as well as diagrams of the ESR transition energies (Figures D-2 and D-4) for each radical. The ESR transition energy diagrams are useful in determining the theoretical positions of ESR absorption lines for a given operating frequency.

Energy levels for ·H (s = 1/2, I = 1/2):

$$E_{(1\,2,\,1\,2)} = 1/2\ g\beta B + 1/4hA_o - 1/2\ g_n\beta_n B \qquad \text{Eq. D-1}$$

$$E_{(1\,2,\,-1\,2)} = 1/2\ [\ (g\beta + g_n\beta_n)^2B^2 + h^2A_o^{\,2}\ ]^{1\,2} - 1/4\ hA_0 \qquad \text{Eq. D-2}$$

$$E_{(-1\,2,\,-1\,2)} = -1/2\ g\beta B + 1/4hA_o + 1/2\ g_n\beta_n B \qquad \text{Eq. D-3}$$

$$E_{(-1\,2,\,1\,2)} = -1/2\ [\ (g\beta + g_n\beta_n)^2B^2 + h^2A_o^{\,2}\ ]^{1\,2} - 1/4\ hA_0 \qquad \text{Eq. D-4}$$

Energy levels for Fremy's salt (s = 1/2, I = 1):

$$E_{(1\,2,\,1)} = 1/2\ g\beta B + 1/2hA_o - g_n\beta_n B \qquad \text{Eq. D-5}$$

$$E_{(1\,2,\,0)} = 1/2\ g_n\beta_n B - hA_o/4 + 1/2[(g\beta B - g_n\beta_n B)(g\beta B - g_n\beta_n B + hA_o) + 9/4\ h^2A_o^{\,2}\ ]^{1/2} \qquad \text{Eq. D-6}$$

$$E_{(1\,2,\,-1)} = -1/2\ g_n\beta_n B - hA_o/4 + 1/2[(g_n\beta_n B - g\beta B)(g_n\beta_n B - g\beta B + hA_o) + 9/4\ h^2A_o^{\,2}\ ]^{1/2} \qquad \text{Eq. D-7}$$

$$E_{(-1\,2,\,-1)} = -1/2\ g\beta B + 1/4hA_o + g_n\beta_n B \qquad \text{Eq. D-8}$$

$$E_{(-1\,2,\,0)} = 1/2\ g_n\beta_n B - hA_o/4 - 1/2[(g_n\beta_n B - g\beta B)(g_n\beta_n B - g\beta B + hA_o) + 9/4\ h^2A_o^{\,2}\ ]^{1\,2} \qquad \text{Eq. D-9}$$

$$E_{(-1\,2,\,1)} = -1/2\ g_n\beta_n B - hA_o/4 - 1/2[(g\beta B - g_n\beta_n B)(g\beta B - g_n\beta_n B + hA_o) + 9/4\ h^2A_o^{\,2}\ ]^{1\,2} \qquad \text{Eq. D-10}$$

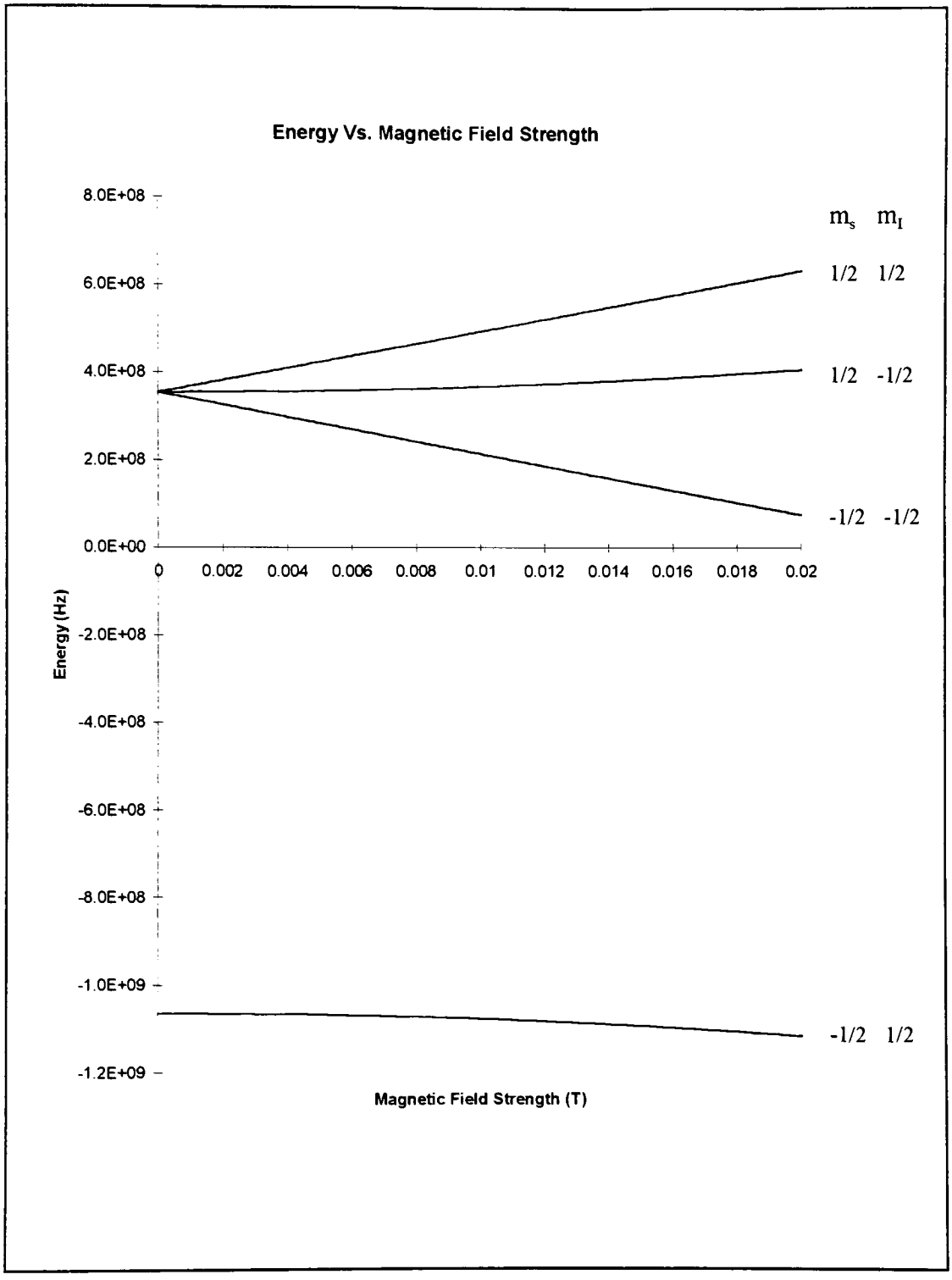


Figure D-1 An energy level diagram for the ·H radical. The four energy levels are generated from Equation D-1 through D-4.

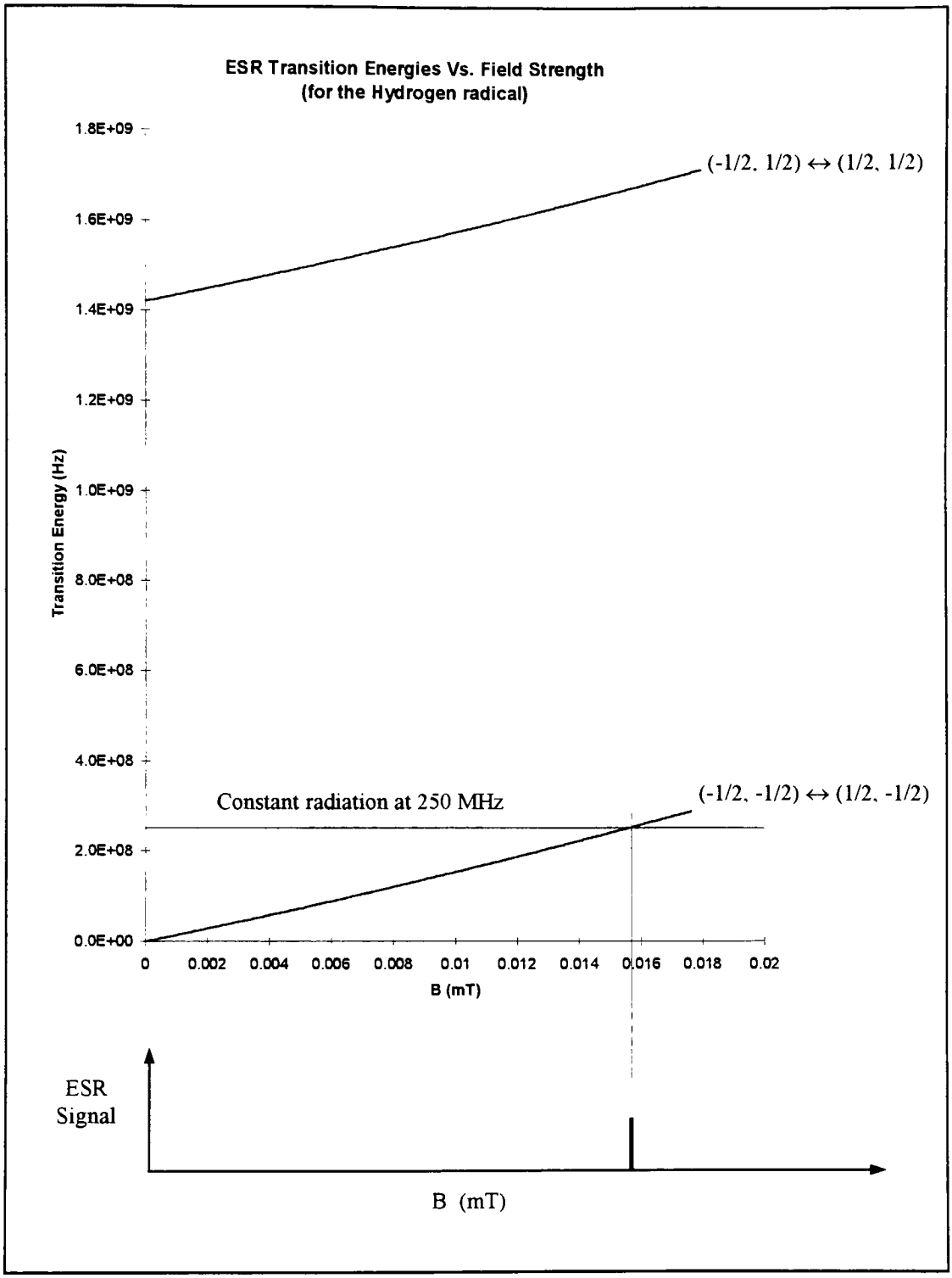


Figure D-2 A plot of ESR transition energies vs. field strength, for the ·H radical. At an operating frequency of 250 MHz, only one ESR absorption line is possible, as shown, and occurs at 15.6 mT.

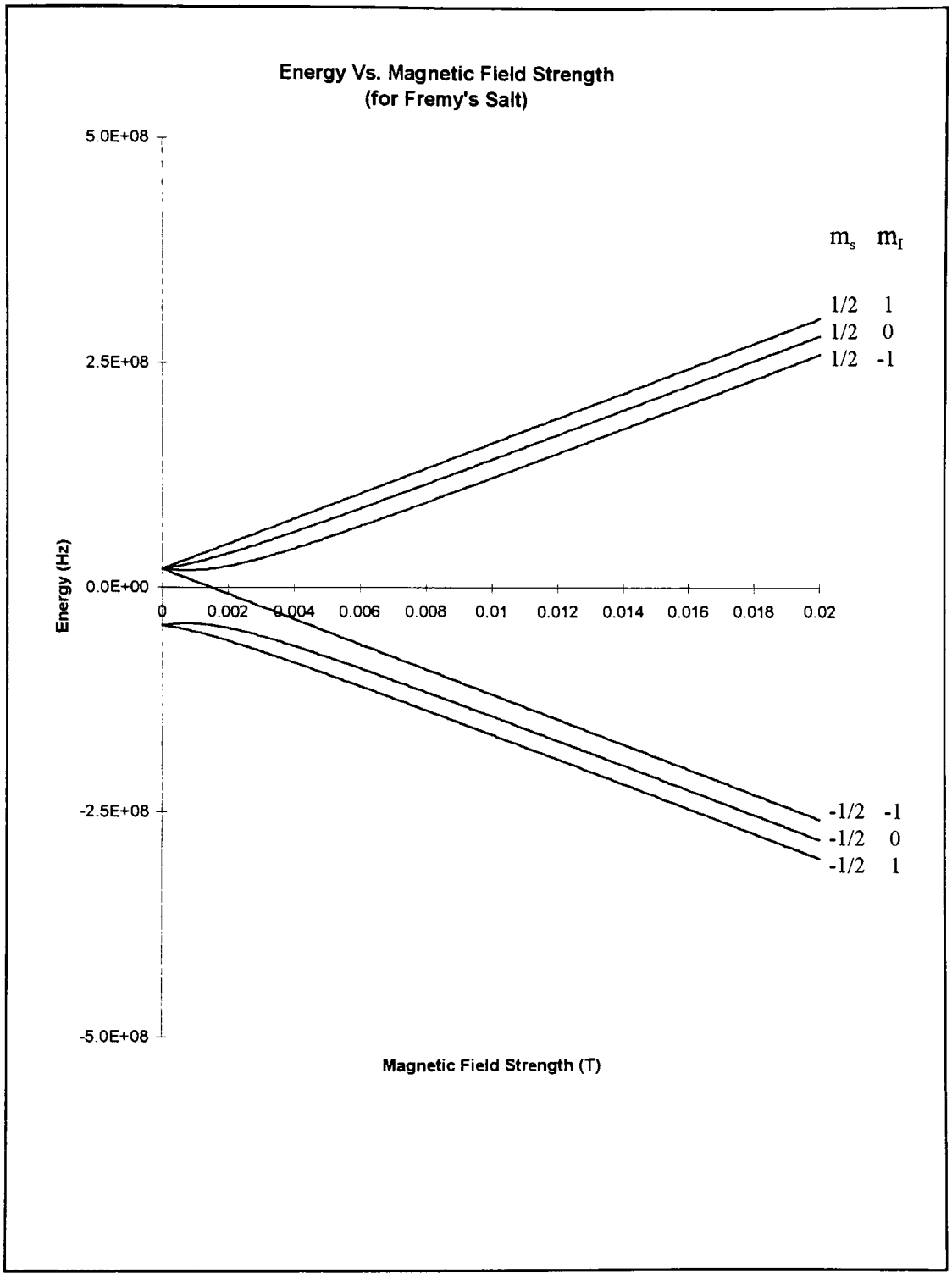


Figure D-3 An energy level diagram for Fremy's salt. The four energy levels are generated from Equation D-5 through D-10.

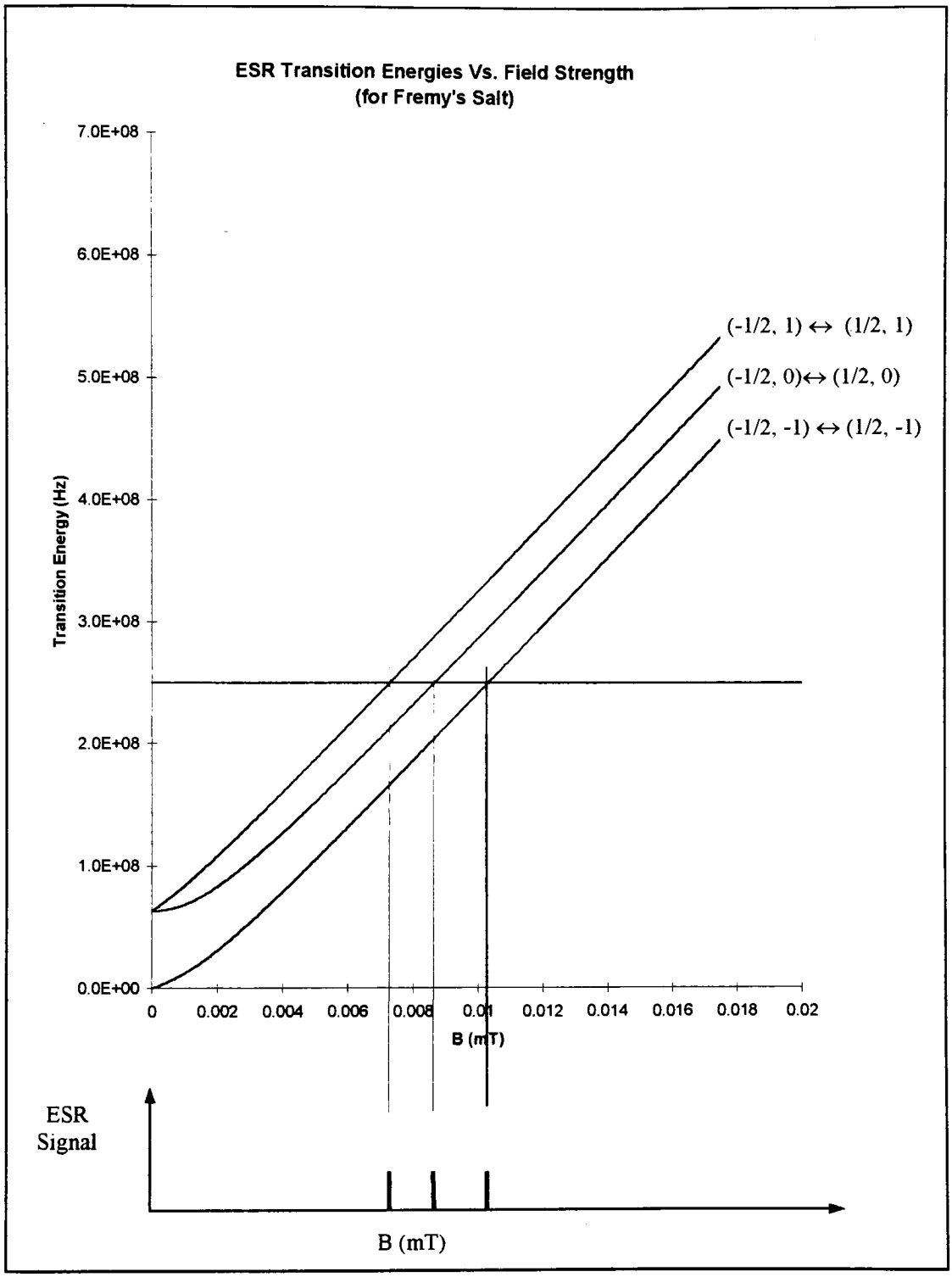


Figure D-4 A plot of ESR transition energies vs. field strength, for Fremy's salt. At an operating frequency of 250 MHz. Three ESR absorption lines are possible, as shown, and occur at 7.3, 8.7 and 10.3 mT.

Appendix E - Basic ESR Instrumentation.

The basic layout of an ESR spectrometer is given in Figure E-1. This is a simplified schematic representation, and is intended to give an overview of the major instrumental subsystems of an ESR spectrometer. The following pages will describe each of the subsystems illustrated in Figure E-1.

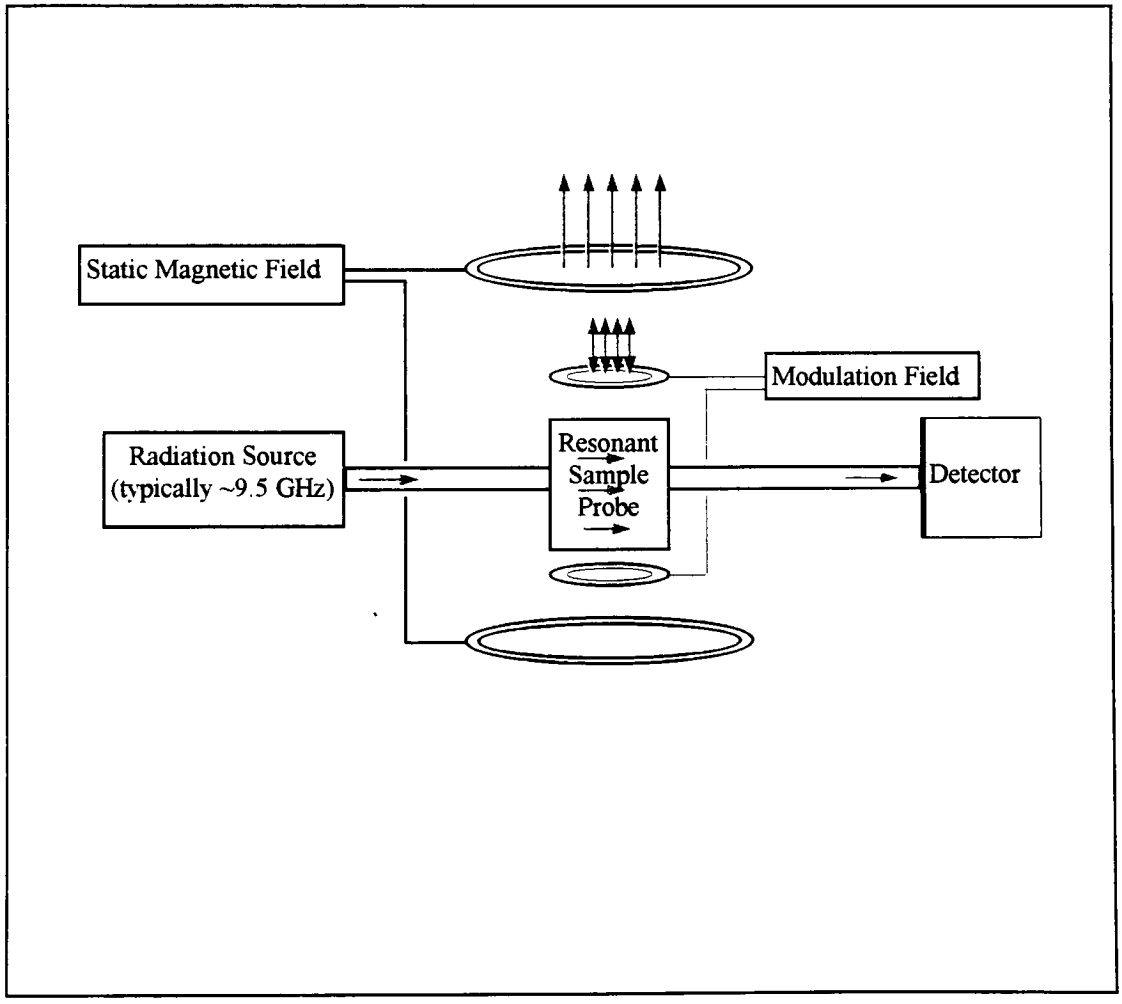


Figure E-1 The instrumentation subsystems of a typical ESR spectrometer.

Typical Source Components.

ESR spectrometers typically operate at a frequency of about 9.5 GHz. This frequency falls within the microwave region of the electromagnetic spectrum. The radiation is usually supplied by a special type of vacuum tube called a klystron. The klystron can be operated in several modes, each of which corresponds to a limited range of frequencies. The output frequency of a klystron can be varied over the operating range of a mode, by varying the voltage supplied to the klystron. This allows the klystron to be tuned to the resonant frequency of the sample cavity. The voltage supplied to the klystron is regulated by an automatic frequency control subsystem, to maintain frequency stability. The automatic frequency controller locks the frequency of the klystron to the resonant frequency of the sample cavity. Since the output frequency of the klystron can be perturbed by reflected microwave energy, an isolator is placed at the output of the klystron. The isolator allows microwave energy to pass freely from the klystron, but strongly attenuates any energy that gets reflected back at the klystron. Reflections can be caused by changes in impedance felt by the microwave energy as it travels through the instrumental components. Typically, a wavemeter is placed after the isolator, and is used in conjunction with a crystal detector to measure the output frequency of the klystron. The power of the microwave energy is controlled by an attenuator. A schematic representation of the Source components of a typical ESR spectrometer is given in Figure E-2.

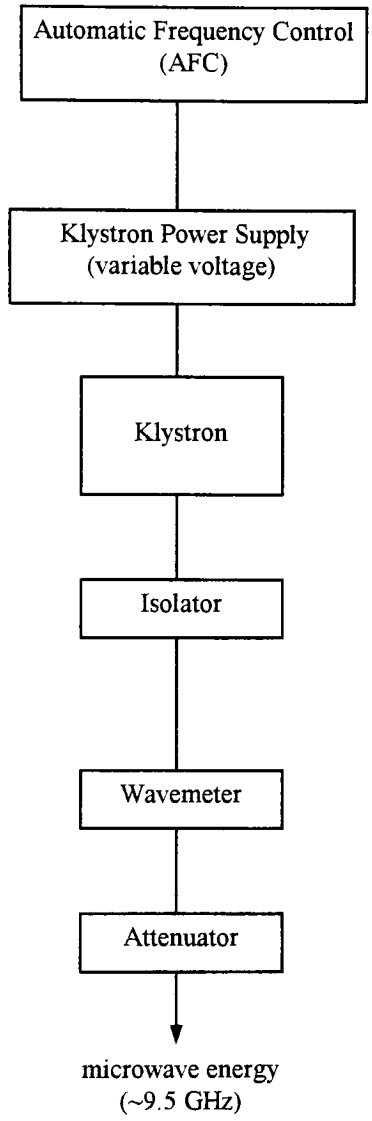


Figure E-2 An illustration of the typical Source components of an ESR spectrometer.

Typical Sample Cavity Components.

ESR absorption of microwave energy involves the interaction of the magnetic component, B_1 , of the microwave energy, with the magnetic dipoles of a sample. For an effective interaction between B_1 and the magnetic dipoles of a sample, the direction of B_1 must be perpendicular to the magnetic dipoles, (i.e. perpendicular to the direction of the applied magnetic field, B). Furthermore, there needs to be a high energy density of the B_1 component of the microwave in the region of the sample. These requirements are fulfilled by utilizing a resonant sample cavity. In a tuned circuit, the cavity is resonant with the microwave energy, allowing the formation of standing waves within the cavity. This results in a sufficiently high energy density of B_1 in the region of the sample. The geometry of the cavity is such that B_1 is perpendicular to the direction of B.

A device called a hybrid tee is used to direct the microwave energy of the source toward the sample cavity. The microwave energy is coupled to the cavity through an adjustable iris. Any impedance mismatch that exists between the transmission line and the cavity will cause a portion of the microwave energy to be reflected back from the cavity, and redirected by the hybrid tee toward the detector. The iris is adjusted to minimize the normally occurring impedance mismatch, thus enhancing the fractional change in reflected energy due to the relatively small change in cavity impedance that arises from ESR absorption of B_1.

Another device used to minimize reflected microwave energy that results from non ESR sources, is called a "slide-screw tuner", and is located between the hybrid tee and the iris. The slide-screw tuner is used to minimize standing waves in the transmission line that result from discontinuities between various line elements. The slide-screw tuner functions by setting up standing waves of the right amplitude and phase to destructively interfere with the unwanted standing waves. A schematic of the cavity system components is given in Figure E-3.

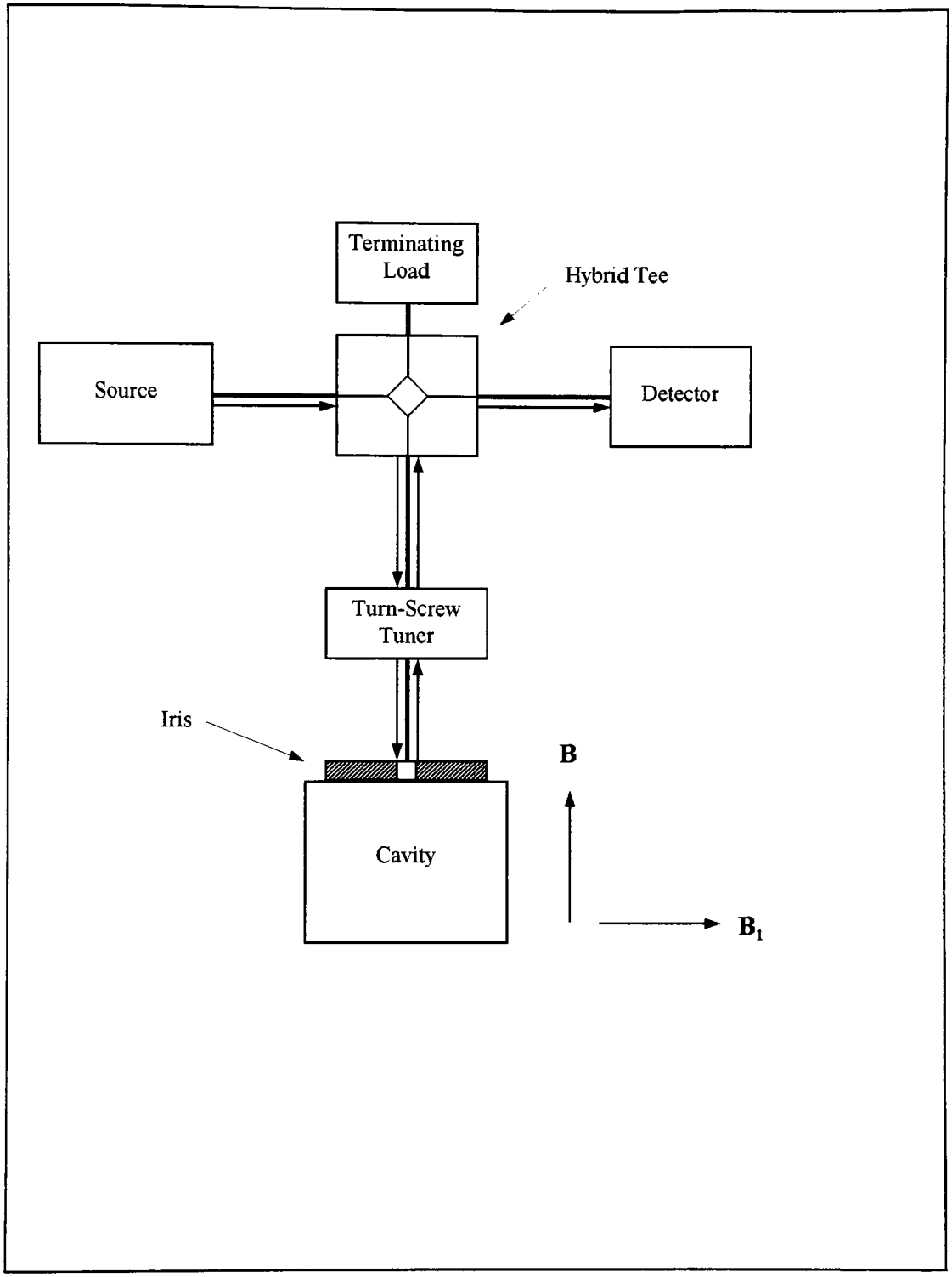


Figure E-3 A schematic representation of the sample cavity system of a typical ESR spectrometer.

Typical Magnet Components.

The magnet of an ESR spectrometer is typically an iron core electromagnet. The magnet must be capable of producing a homogeneous static field in the region of the sample, and must be linearly adjustable. Homogeneity of the magnetic field should be kept within a tolerance of $\pm 0.1 \mu T$. The magnet is energized by a current regulated power supply. The magnetic field strength is directly proportional to the current that is delivered by the power supply. A field sweep driver is utilized to linearly vary the output of the magnet power supply. The field sweep driver can consist of a programmable control voltage, that is utilized at a control port of the magnet power supply. A schematic of the magnet components is given in Figure E-4.

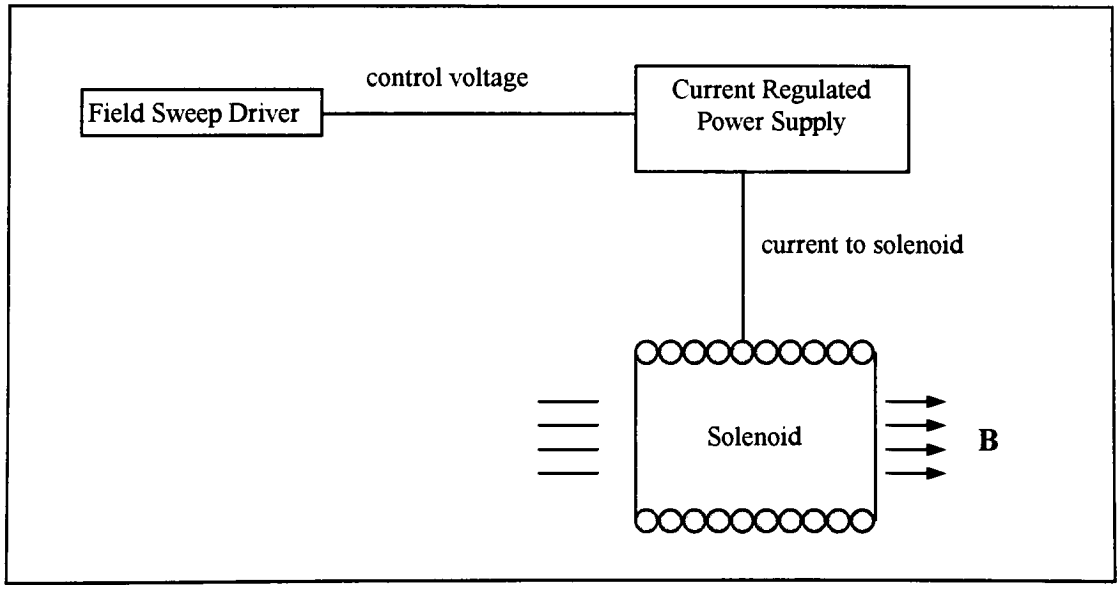


Figure E-4 A schematic representation of the magnet system of a typical ESR spectrometer.

Typical Field Modulation and Detection Components.

One of the key problems that needs to be addressed in ESR spectroscopy, is the large signal-to-noise ratio (SNR). The relatively small fractional amount of microwave energy that is seen by the detector as a result of ESR absorption, is usually too small to allow for direct signal detection. As a consequence, a "phase-sensitive" detection scheme is typically utilized to detect ESR signals. Phase sensitive detection works by modulating the ESR signal at a selected frequency. The modulated ESR signal is selectively amplified and filtered, limiting noise contributions to frequencies within a narrow band surrounding the frequency of the modulated signal. This usually results in a sufficient enhancement of the SNR for favorable signal detection to occur.

The ESR signal is modulated by placing Helmholtz coils on either side of the sample cavity, along the axis of the applied magnetic field. The Helmholtz coils are typically energized by 100 kHz RF energy. The modulating field that is induced by the Helmholtz coils is superimposed on the applied static magnetic field, causing the applied static magnetic field to experience small amplitude modulations, at 100 kHz. The detector current resulting from ESR absorption will modulate at 100 kHz, and have an amplitude that is directly related to the difference in ESR absorption at the maximum and minimum values of a given modulated applied magnetic field strength (See Figure E-5a). Thus, the rectified detector current represents the first derivative of the ESR absorption with respect to field modulation amplitude (See Figure E-5b).

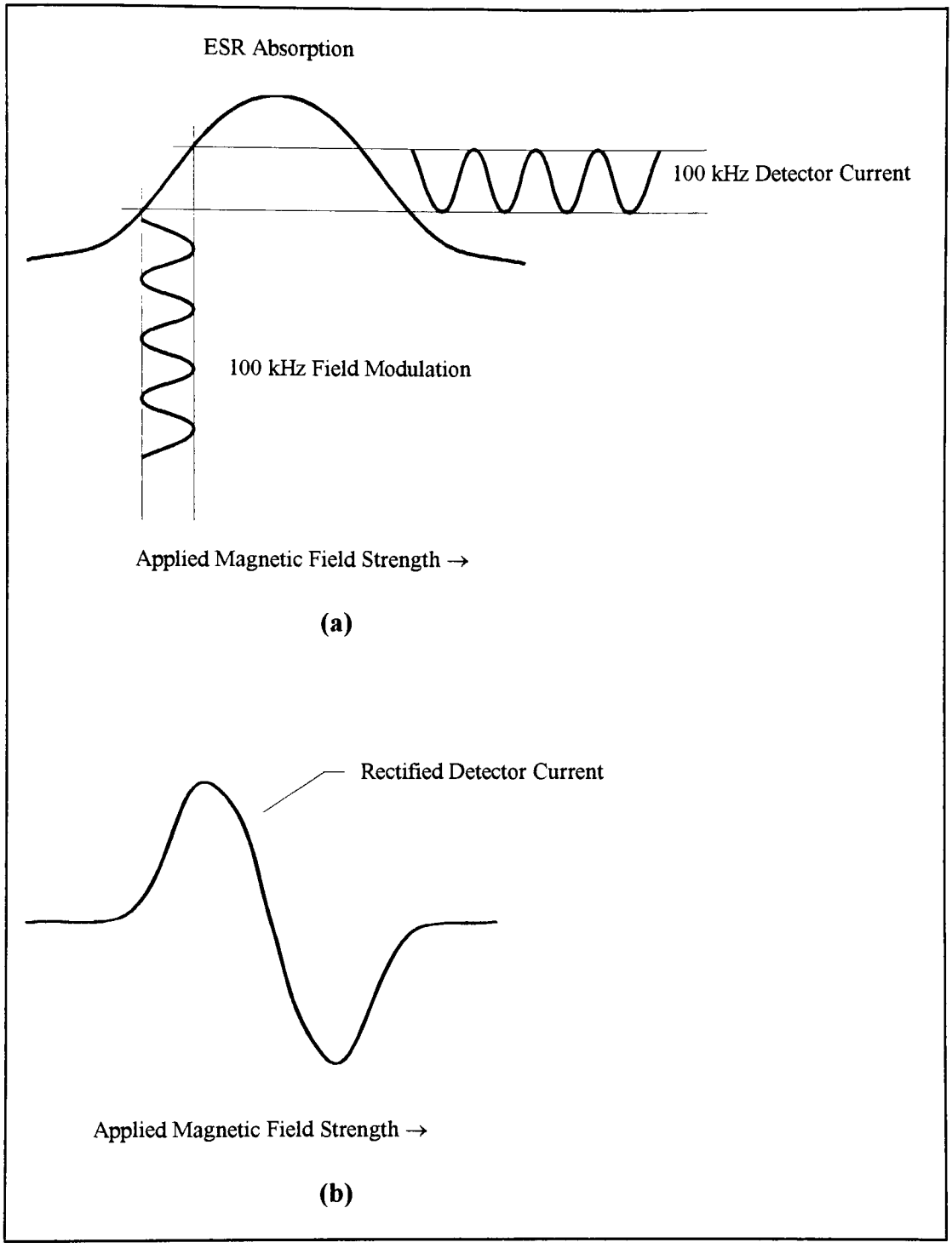


Figure E-5 Phase sensitive detection. **(a)** The effect of small amplitude field modulation on the applied magnetic field, and the resulting detector current. **(b)** The first derivative ESR spectrum that is obtained by using phase sensitive detection.

The modulation subsystem starts at a 100 kHz signal generator. A portion of the low power output of the 100 kHz signal generator is sent to a power amplifier. The power amplifier boosts the power of the 100 kHz signal to a level that is appropriate to energize the Helmholtz modulation coils with. The level of the modulation field produced by the coils is controlled by adjusting the output of the power amplifier. The heart of the detection subsystem is a silicon crystal detector. The crystal detector rectifies the microwave power that it receives from the hybrid tee. The output current of the crystal detector varies as a function of the amount of microwave power it receives. The output of the detector will be the 100 kHz modulated ESR signal. The 100 kHz modulated ESR signal is amplified and sent to a lockin amplifier. Within the lockin amplifier, narrow bandpass filters are used to discard any noise components, except for those that are very close to the 100 kHz modulation frequency of the ESR signal. The lockin amplifier utilizes high gain amplification to enhance the modulated ESR signal, and demodulates the ESR signal by mixing it with a phase-shifted 100 kHz reference signal in a device called a product detector. The output of the lockin amplifier is the demodulated ESR signal, which is monitored and displayed as a function of the applied field strength, B. Figure E-6 shows a schematic of the modulation and detection components of a typical ESR spectrometer.

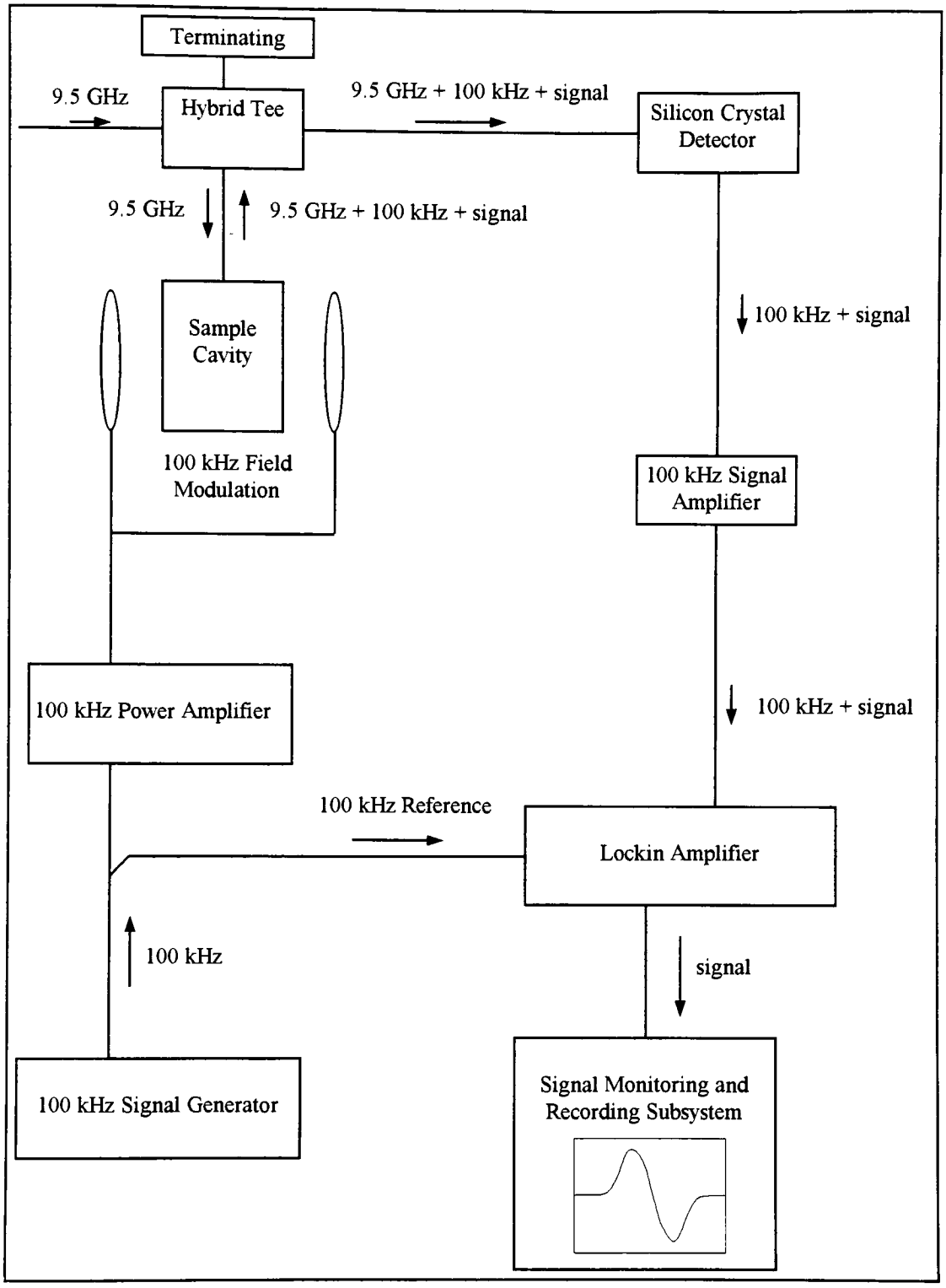
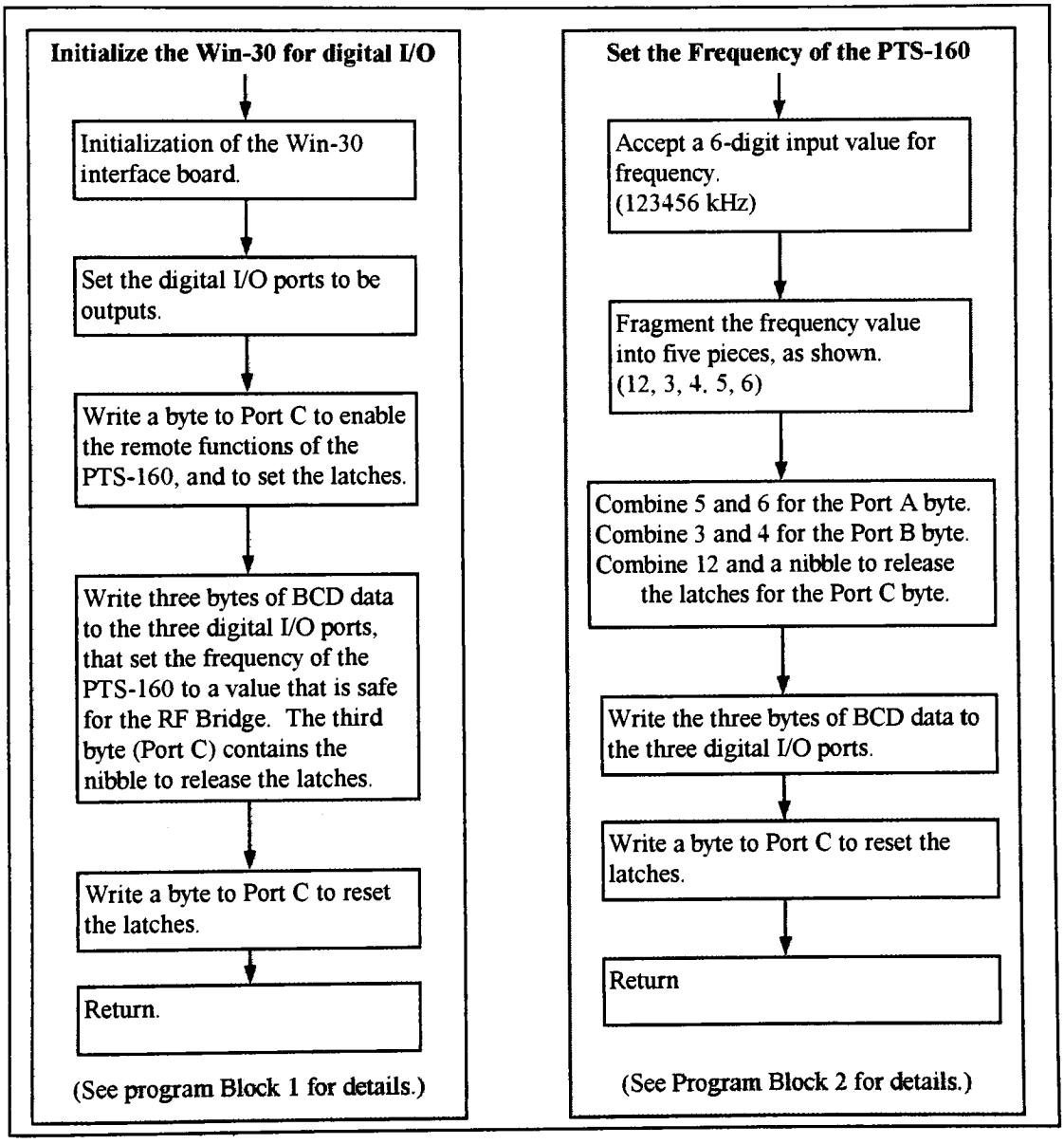


Figure E-6 A schematic representation of the modulation and detection subsystem of a typical ESR spectrometer.

Appendix F - Software Development

Software control of the digital interface between the Win-30 and the PTS-160 is illustrated in Flow Chart F-1 and Program Blocks F-1 and F-2.



Flow Chart F-1 Flow charts of the C++ functions that were written to allow the frequency of the PTS-160 to be set through the program connector of the PTS-160.

This initializes the Win-30 board, and sets the state of each digital I/O port for digital output.

NOTE:
The set_base(), diag(), init(), and d_mode() Win-30 C++ driver functions are located on the UEIDAQ.LIB file, which was be included as a header file in this this program.

```
int diag_err;
initialize_win_30( )
{
        set_base(base);
        diag_err = diag();
        if (diag_err != 0)
        {
                cout<<"ERROR (WIN-30 faulty or not found)\n";
                return 0;
        }
        init( );
        d_mode(0,0,0);          //sets the 3 I/O ports to be outputs (mode=0)
```

This sets the remote enable bit high, and all three latches low

```
        d_out(2,67);
```

This sequence of commands writes the three bytes of data to the PTS-160. However, the values of the digit modules are not updated to result in the new frequency, until the third and final byte is sent. The upper nibble of the third byte contains the data to simultaneously reset all three latches to low, allowing the simultaneous update of the digit values.

```
        d_out(0,255);  Sets the PTS to 124000 kHz (a safe value for the LFESR)
        d_out(1,191);
        d_out(2,115);  a and b latches set low, c is low
```

This resets the latches for Port A and B bytes to high before returning to the main program. This prevents transient frequency values from occurring during following frequency value update operations.

```
        d_out(2,67);
        return 0;
}
```

Program Block F-1 This section of C++ code was written to initialize the Win-30 for digital output operations, and to preset and lock the frequency of the PTS-160 to a value of 248 MHz.

```
void set_pts(long int frequency)
{
        long int freq;
        long int _10MHzDigit;
        long int _1MHzDigit;
        long int _100KHzDigit;
        long int _10KHzDigit;
        long int _1KHzDigit;

        long int A_low;
        long int A_high;
        long int B_low;
        long int B_high;
        long int C_low;
        long int C_high;

        long int A;
        long int B;
        long int C;

        This assigns the value of frequency to a temporary variable called freq,
        that can be operated on without destroying the current value stored in
        frequency.

        freq = frequency;

        This breaks the 6-digit variable into six separate variables.

        _10MHzDigit = freq/10000;
        _1MHzDigit = (freq/1000)-(freq/10000*10);
        _100KHzDigit = (freq/100)-(freq/1000*10);
        _10KHzDigit = (freq/10)-(freq/100*10);
        _1KHzDigit = (freq)-(freq/10*10);
```

continued ...

Program Block F-2 (p. 1 of 3) This C++ function was written to set the frequency of the PTS-160 to the value stored in the frequency variable. This allows the frequency to be readily set to a given value, from within frequency sweeping loops.

The values stored in the six separate digit variables, are stored in variables that correspond to the 6 nibbles of data that will be transmitted by the digital I/O ports of the Win-30 to the PTS-160.

A_low = _1KHzDigit;
A_high = _10KHzDigit;
B_low = _100KHzDigit;
B_high = _1MHzDigit;
C_low = _10MHzDigit;
C_high = 8; **(This nibble is used for setting the remote enable bit and the three latch bits.)**

The 6 separate nibbles are combined to form 3 separate bytes. This is accomplished by multiplying the value of each high nibble nibble by 4 orders of 2, and adding the result to the corresponding low nibble.

A = 255 - (A_low + 16*A_high);
B = 255 - (B_low + 16*B_high);
C = 255 - (C_low + 16*C_high);

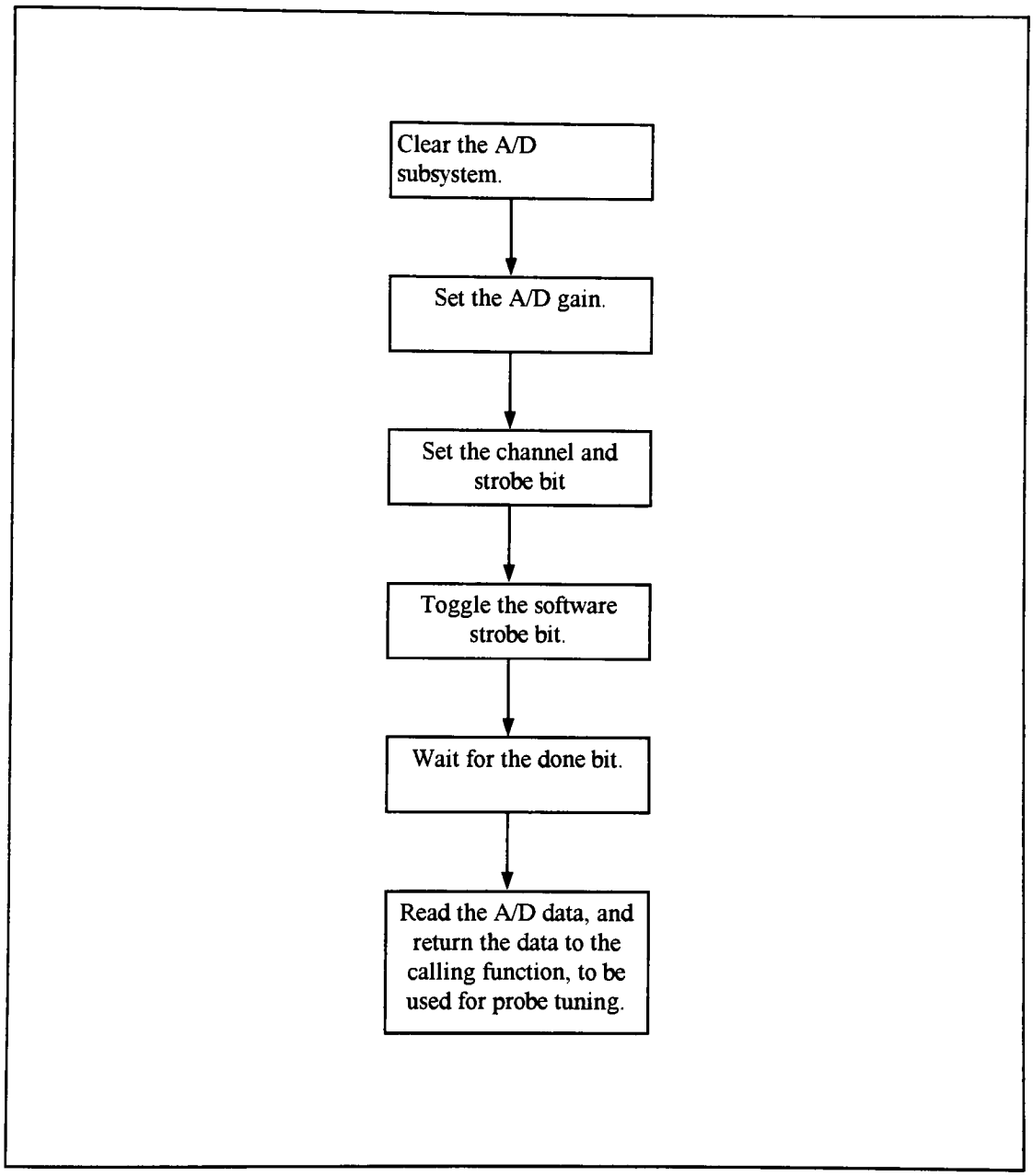
Continued ...

Program Block F-2 (p. 2 of 3) This C++ function was written to set the frequency of the PTS-160 to the value stored in the frequency variable. This allows the frequency to be readily set to a given value, from within frequency sweeping loops.

```
        This section of code uses the _outp(address, byte) function to write the
        three bytes of data to the PTS-160, using the three digital I/O ports of
        the Win-30. The hexadecimal addresses of ports A, B and C, are 708,
        709 and 70A, respectively.
        The high nibble of Port C sets all three latches low, causing all five of the
        digit modules to be updated simultaneously.

        _outp(0x708,(int)A);
        _outp(0x709,(int)B);
        _outp(0x70A,(int)C);

        This resets the latches for Port A and B bytes to high, preventing
        transient frequency values from occuring durring following frequency
        value update operations.

        C_high = 11;
        C = 255 - (C_low + 16*C_high);
        _outp(0x70A,(int)C);

return ;

}
```

Program Block F-2 (p. 3 of 3) This C++ function was written to set the frequency of the PTS-160 to the value stored in the frequency variable. This allows the frequency to be readily set to a given value, from within frequency sweeping loops.

Implementation of the A/D interface required the register-level development of software algorithms to perform the tasks indicated in Flow Chart F-2. Details of the C++ code, are given in Program Block F-3.



Flow Chart F-2 This shows the flow of the A/D interface software algorithms.

This function clears the A/D subsystem of the Win-30 Interface Board.

```
clear_ad_subsystem()
{
        _outp(admde,0x92);
        _outp(adccr,0x2);
        _inpw(addatl);

        return 0;
}
```

This function sets the gain on all A/D channels to zero.

```
set_gain()
{
        outp(gmem0,0);
        outp(gmem1,0);
        outp(gmem2,0);
        outp(gmem3,0);

        return 0;
}
```

This function sets the channel to be converted (channel 0), and sets the strobe select bit of the A/D subsystem.

```
set_channel_and_strobe_select()
{
        _outp(adccr,2);    //set the channel to be ch0, and set the strobe select

        return 0;
}
```

continued ...

Program Block F-3 (p. 1 of 2) This section of C++ code was written to digitize the voltage that results at the detector diode, D1, of the Modulation and Detection subsystem. This allows the frequency response of the STS sample probe to be monitered from within frequency sweeping loops, durring probe-tuning operations.

This function toggels the software strobe bit.

```
toggel_the_software_strobe_bit()
{
        _outp(adccr,3);   //toggel the software strobe bit to "high"
        _outp(adccr,2);   //toggel the software strobe bit to "low"

        return 0;
}
```

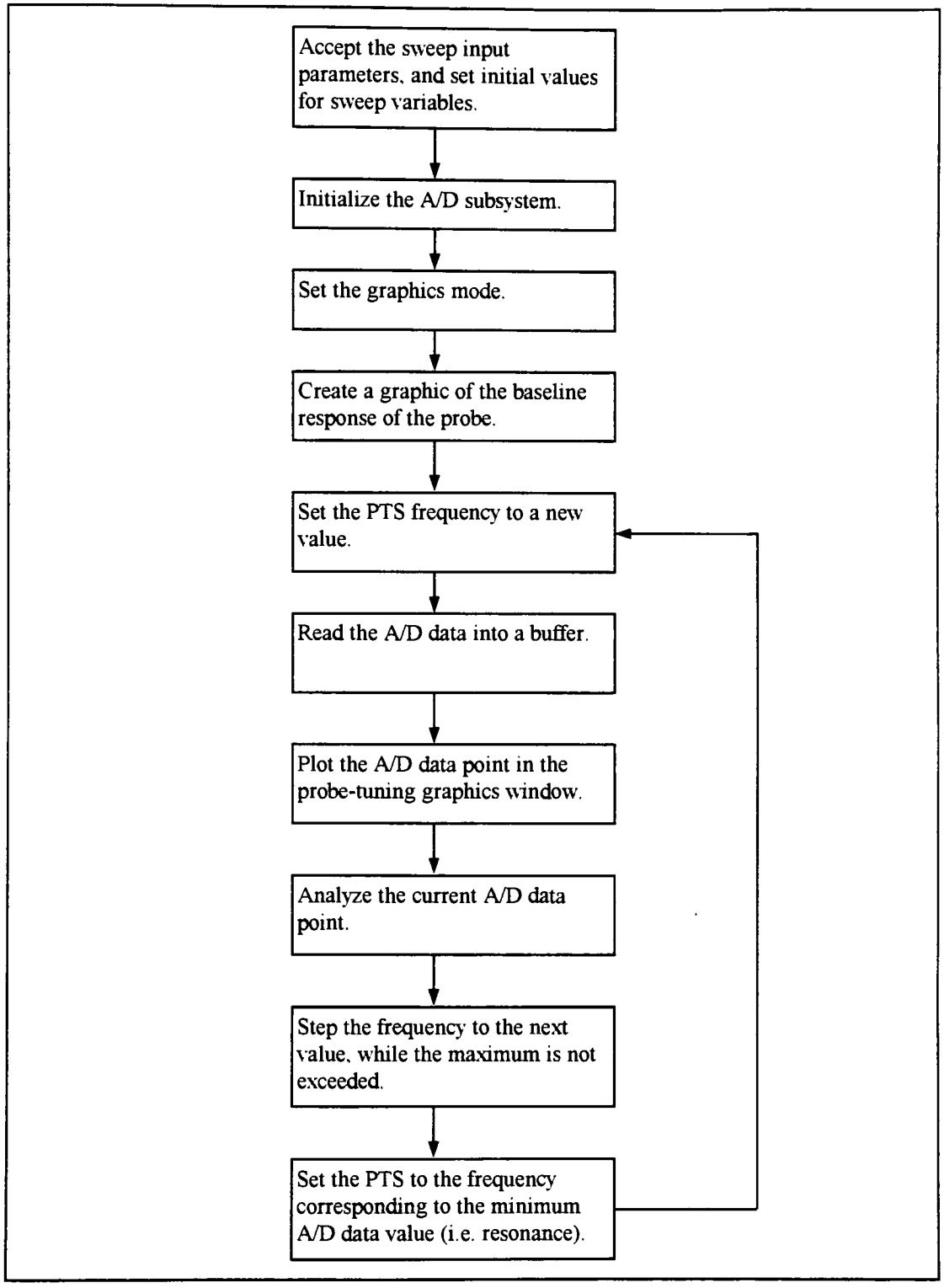
This function waits for the done bit to be set by the Win-30 board.

```
wait_for_done_bit()
{
        float done_bit;
        done_bit=_inp(admde)&64;
        do
        {
                done_bit=_inp(admde)&64;
        }
        while (done_bit!=64);

        return 0;
}
```

This function reads the A/D data from the data register after the done bit is set, and returns the digized data to the calling function, to be used for probe-tuning purposes.

```
short read_ad_data()
{
        short y;
        float     word;
        word = (_inpw(addatl)&4095);
        word = (word-2048)*5/2048;
        word = 200.0f * word;
        y = (short)word;

        return y;
}
```

Program Block F-3 (p. 2 of 2) This section of C++ code was written to digitize the voltage that results at the detector diode, D1, of the Modulation and Detection subsystem. This allows the frequency response of the STS sample probe to be monitored from within frequency sweeping loops, during probe-tuning operations.



Flow Chart F-3 This shows the flow of the automatic probe tuning software. See Program Block F-4 for details of the corresponding C++ code.

This function plots the A/D data in a probe tuning graphics window.

```
void tuning_plot(short x,short y, short yy)
{
        float tune_x_step = (float)200 / (float)(width/freq_step);

        x=(short)(freq_step_counter * tune_x_step + 20);
        y=y+20;
        yy=yy+20;

        //_rectangle(_GBORDER, x-1,y-1,x+1,y+1);
        //_rectangle(_GFILLINTERIOR, x-1,yy-1,x+1,yy+1);

        _setpixel( x,y);
        _setpixel( x,yy);

        return;
}
```

This section of code performs the frequency sweep and the data analysis for the resonant frequency of the probe.

```
long int sweep()
{
        long int resonance_freq;
        short voltage_minimum = 9999;
        short x,y,yy;
        set_pts_initial();

        //Calculate the scan-loop parameters

        long int low_freq, high_freq, temp_x;

        //*********************************************************
        // This divides the sweep input parameters before setting the PTS,
        // since the output of the PTS goes to a frequency doubler.
                center_freq = center_freq / 2;
                width = width / 2;
                freq_step = freq_step / 2;

                low_freq = center_freq - (width/2);

                high_freq = center_freq + (width/2);
        //*********************************************
```

continued ...

Program Block F-4 (p. 1 of 5) This section of C++ code was written to perform automatic probe tuning operations for the LFESR operating system.

```
        //Initialize the A/D subsystem.

        clear_ad_subsystem();
        set_gain();
        set_channel_and_strobe_select();


        /* Find a valid graphics mode. */
        if( !_setvideomode( _MAXRESMODE ) )
    exit( 1 );
        _rectangle( _GBORDER, 20, 20, 220, 220 );
        //*********************************************
                frequency = 1;
                set_pts(frequency);
                                delay(50);
                                toggel_the_software_strobe_bit();
                                wait_for_done_bit();
                                y = read_ad_data();
                                yy = y;              // yy = the base-line voltage.
              //*********************************************




                frequency = low_freq;
                freq_step_counter = 0;

                do
                {
                        set_pts(frequency);
                        delay(60);
                        toggel_the_software_strobe_bit();
                        wait_for_done_bit();
                        y = read_ad_data()*tuning_gain;
                        temp_x = (frequency - low_freq)/2;
                        x = (short)temp_x;
                        tuning_plot((short)freq_step_counter,y,yy);
                        probe_tune_data[freq_step_counter] = y-yy;

                                                if (y < voltage_minimum)
                                                {
                                                voltage_minimum = y;
                                                resonance_freq = frequency;
                                                }
```

continued ...

Program Block F-4 (p. 2 of 5) This section of C++ code was written to perform automatic probe tuning operations for the LFESR operating system.

```
                    frequency = frequency + freq_step;
                    freq_step_counter = freq_step_counter+1;
                }
                while (frequency <= high_freq);

        frequency = resonance_freq;

        //*********************************************************
        // This restores the sweep parameters, after using them to set the PTS.
        // which allows the parameter output values to reflect the un-divided
        // values.

                center_freq = center_freq * 2;
                width = width * 2;
                freq_step = freq_step * 2;

        //*********************************************

        return frequency;
}
```

This section of code calls the frequency sweep loop for as many cycles as the operator has requested. This section also performs data saving operations, by streaming the data buffer to a user-defined disk file, when requested.

```
tune_probe()
{

        /* Find a valid graphics mode. */
        if( !_setvideomode( _MAXRESMODE ) )
   exit( 1 );
        _rectangle( _GBORDER, 20, 20, 220, 220);



        int counter = 1;
        do
        {
        frequency = sweep();

        counter ++;
        }
        while (counter <= num_of_tuning_sweeps);

        cout<<"Resonance Frequency of Probe = "<<2*frequency <<"\n";
```

continued ...

Program Block F-4 (p. 3 of 5) This section of C++ code was written to perform automatic probe tuning operations for the LFESR operating system.

```
set_pts(frequency);

//*************************************************************************
//This section allows the user to save the probe tuning data to a disk file

                ofstream fstr;
        int     choice = 0;
        char choice_input = 0;


        do
        {
                cout<<"Save the probe-tuning sweep data ? \n";
                yes_no_menu();
                cin >> choice_input;
                choice = (int) choice_input;
        }
        while (((int)choice < 49) || ((int)choice > 50));



        switch(choice)
        {



                case    49:
                        //save data code
                        char file_name[12];
                        cout << "Enter a drive, filename, and extension\n";
                        cin >> file_name;

                        fstr.open(file_name, ios::out);
                        if(!fstr.good())
                                cout << "ERROR opening file.\n";
                                else
                                {
                                        fstr << "Center Frequency (kHz) = " <<
center_freq << "\n";
                                        fstr << "Sweep Width (kHz)     = " << width
<< "\n";
                                        fstr << "Step Size (kHz)       = " << freq_step
<< "\n";
                                        fstr << "Resonant Freq (kHz)   = " <<
frequency << "\n";
```

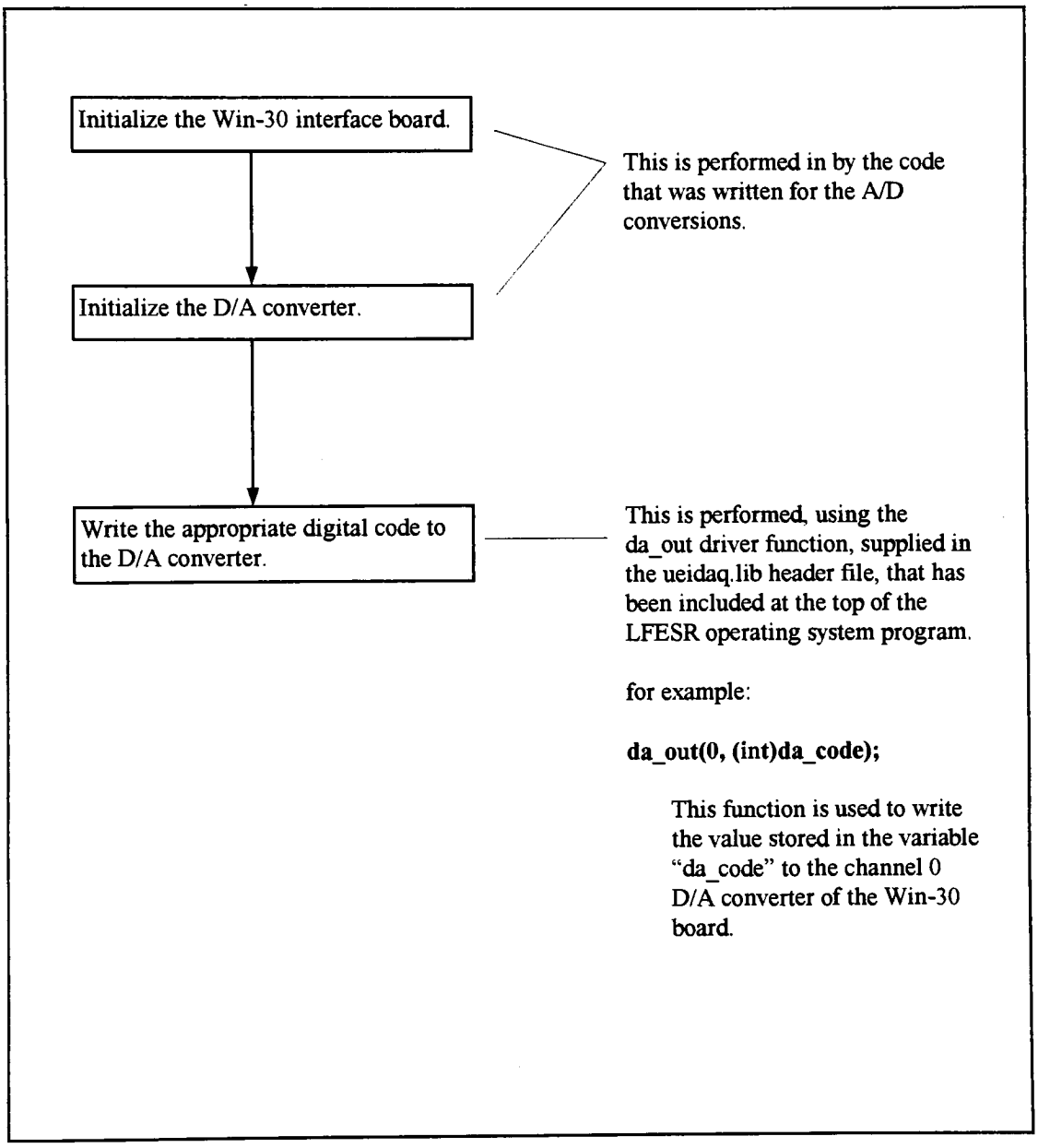
continued ...

Program Block F-4 (p. 4 of 5) This section of C++ code was written to perform automatic probe tuning operations for the LFESR operating system.

```
                                        int ctr = 0;
                                        for(ctr = 0; ctr <= freq_step_counter-1; ctr++)
                                        {
                                                fstr << probe_tune_data[ctr] <<"\n";
                                        }
                                }


                        break;




                case    50:
                        //
                        break;

        }

    return 0;
}
```

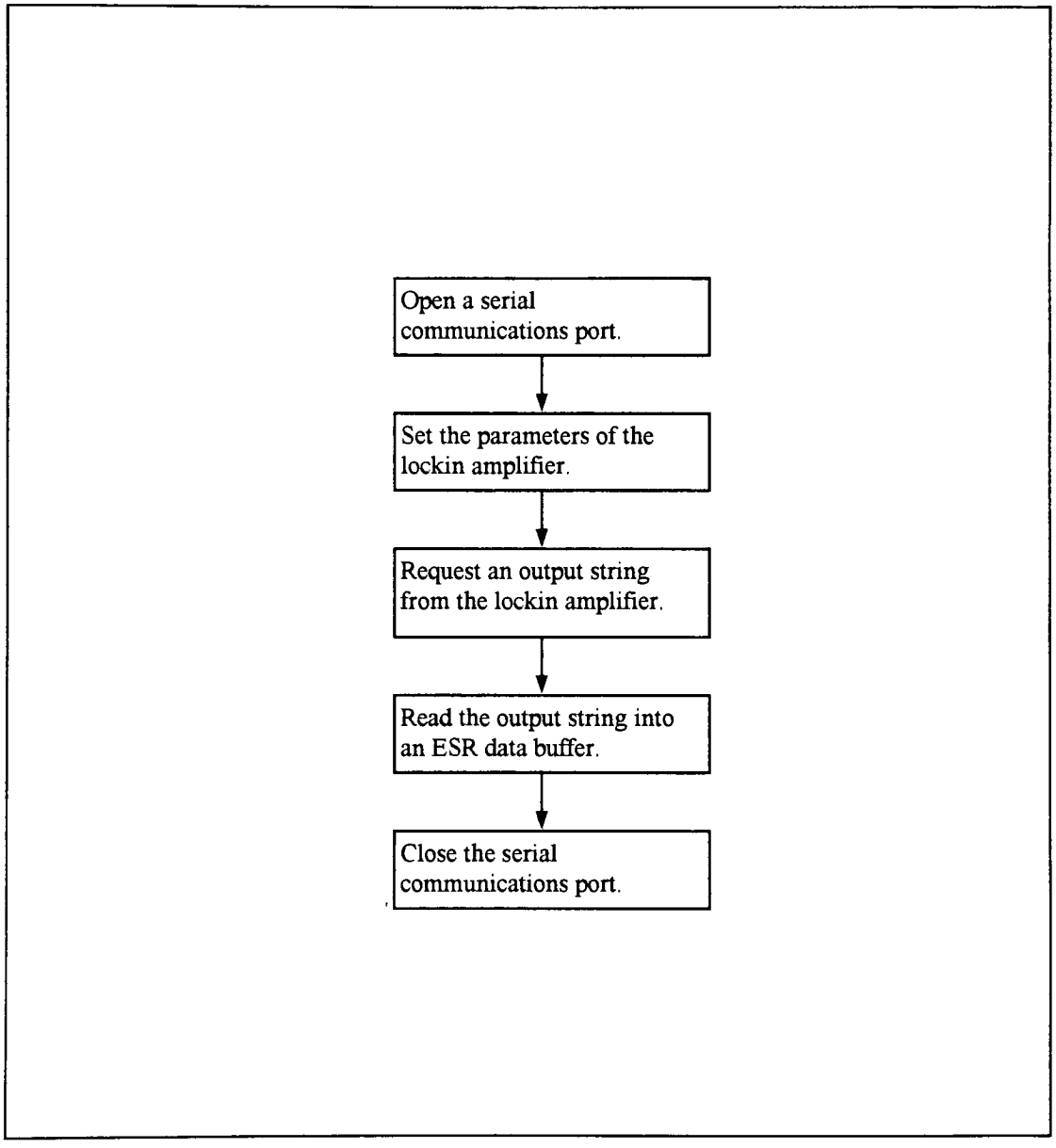
Program Block F-4 (p. 5 of 5) This section of C++ code was written to perform automatic probe tuning operations for the LFESR operating system.

The magnet is software controlled via the D/A interface. Flow Chart F-4 illustrates the flow of the software functions for the magnetic field strength to be set to a given value.



Flow Chart F-4 This shows the flow of the software that is used to set the magnetic field strength, using the D/A converter of the Win-30 board.

Flow Chart F-5 illustrates the flow of the software functions for the RS232 interface between the computer and the lockin amplifier. Details of the corresponding C++ code are given in Program Block F-5.



Flow Chart F-5 The flow of software used for controlling the lockin amplifier.

These are variable and string definitions.

```
char character;

int idComDev;
        DCB dcb;

int err;

char lockin_data[16];
```

This command string sets the lockin operational parameters to start-up values, with the sensitivity set to 500 mV (G24).

```
char lockin_string1[61] =
"B1;L1,1;L2,1;D0;P+88;M0;E1,0;E2,0;T1,6;T2,0;G24;AX;AY;S0;I1\r";
```

This command string changes the lockin sensitivity from 500 mV (G24) to 10 μV (G10), leaving all other parameters at the start-up values.

```
char lockin_string2[61] =
"B1;L1,1;L2,1;D0;P+88;M0;E1,0;E2,0;T1,6;T2,0;G10;AX;AY;S0;I1\r";
```

This command string is used to poll the lockin amplifier for ESR data.

```
char output_string[4] = "Q1\r";
```

This function opens the COM 2 port of the computer, for serial comunications.

```
open_com_2()
{

        typedef struct tagDCB       /* dcb */
        {
          BYTE Id;
```

Program Block F-5 (p. 1 of 4) These sections of C++ code were written to control the lockin amplifier through RS232 serial communications.

```
    UINT BaudRate;
    BYTE ByteSize;
    BYTE Parity;
    BYTE StopBits;
    UINT RlsTimeout;
    UINT CtsTimeout;
    UINT DsrTimeout;

    UINT fBinary      :1;
    UINT fRtsDisable  :1;
    UINT fParity      :1;
    UINT fOutxCtsFlow :1;
    UINT fOutxDsrFlow :1;
    UINT fDummy       :2;
    UINT fDtrDisable  :1;

    UINT fOutX        :1;
    UINT fInX         :1;
    UINT fPeChar      :1;
    UINT fNull        :1;
    UINT fChEvt       .1;

    UINT fDtrflow     :1;
    UINT fRtsflow     :1;
    UINT fDummy2      :1;

    char XonChar;
    char XoffChar;
    UINT XonLim;
    UINT XoffLim;
    char PeChar;
    char EofChar;
    char EvtChar;
    UINT TxDelay;
} DCB;
```

Program Block F-5 (p. 2 of 4) These sections of C++ code were written to control the lockin amplifier through RS232 serial communications.

```
        DCB dcb;

        int err;


        idComDev = OpenComm("COM2", 1024, 128);
        if (idComDev < 0)
        {
        cout<<"ERROR (OpenComm) = ("<<err<<")\n";
        return 0;
        }
        else
        {
                cout<<"idComDev = "<<idComDev<<"\n";
        }


        err = BuildCommDCB("COM2:9600,n,8,2", &dcb);
        if (err < 0)
        {
        cout<<"ERROR (BuildCommDCB) = ("<<err<<")\n";
        return 0;
        }


        err = SetCommState(&dcb);
        if (err < 0)
        {
                cout<<"ERROR (BuildCommDCB) = ("<<err<<")\n";
      return 0;
        }


   return 0;
}
```

This function closes the RS232 port.

```
close_com_2()
```

Program Block F-5 (p. 3 of 4) These sections of C++ code were written to control the lockin amplifier through RS232 serial communications.

```
{
   CloseComm(idComDev);

        return 0;

}
```

This section of code is executed at the time the operating program is opened up. It sets the operating parameters of the lockin amplifier, with the lockin sensitivity set to 500 mV (The least sensitive mode). This is so that the lockin is not damaged durring probe tuning operations.

```
        open_com_2();
        WriteComm(idComDev, lockin_string1, 60);
        delay(500);
        close_com_2();
```

This section of code is used to reset the lockin sensitivity from a value of 500 mV to 10 uV for the purpose of reading ESR data durring ESR scan operations.

```
        open_com_2();
        WriteComm(idComDev, lockin_string1, 60);
        delay(500);
```

This section of code is used to request an output string from channel 1 of the lockin, and to read the output string into a data buffer, for use by the LFESR software.

```
        WriteComm(idComDev, output_string, 3);
        delay(250);
        ReadComm(idComDev, lockin_data, 16);
```

This section of code resets the lockin sensitivity to 500 mV, and closes com 2 until the next ESR scan is requested.

```
        WriteComm(idComDev, lockin_string1, 60);
        delay(250);
        close_com_2();
```

Program Block F-5 (p. 4 of 4) These sections of C++ code were written to control the lockin amplifier through RS232 serial communications.